© 2016 Gregory Paul Meyer

REAL-TIME 3D FACE LOCALIZATION AND VERIFICATION

BY

GREGORY PAUL MEYER

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Professor Minh N. Do, Chair
Professor Sanjay J. Patel
Professor John C. Hart
Associate Professor Derek Hoiem

ABSTRACT

We present a method for real-time 3D face localization and verification using a consumer-grade depth camera. Our approach consists of three parts: face detection, head pose estimation, and face verification. Face detection is performed using a standard detection framework which we significantly improve by leveraging depth information. To estimate the pose of the detected face, we developed a technique that uses a combination of the particle swarm optimization (PSO) and the iterative closest point (ICP) algorithm to accurately align a 3D face model to the measured depth data. With the face localized within the image, we can compare a database 3D face model to the depth image to verify the identity of the subject. We learn a similarity metric using a random decision forest to accurately authenticate the subject. We demonstrate state-of-the-art results for both face localization and face verification on standard datasets. Since the camera and our method operate at video rate, our system is capable of continuously authenticating a subject while he/she uses his/her device.

To my family and friends

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor Minh Do for his guidance and support of my research. I would also like to thank my thesis committee, Sanjay Patel, John Hart, and Derek Hoiem, for their advice and knowledge. Thanks to all my fellow lab mates for their help and camaraderie, especially Ben Chidester and Trong Nguyen for allowing me to use their likenesses in many papers and presentations. I would like to recognize my family and friends for all of their love and encouragement. Finally, I would like to express my love and appreciation to Christine Bagtas, whose love, support, and patience was paramount to my experience in graduate school.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

3D	Three Dimensional
RGB	Red, Green, Blue
CPU	Central Processing Unit
GPU	Graphics Processing Unit
ICP	Iterative Closest Point
PSO	Particle Swarm Optimization
PCA	Principle Component Analysis
SVM	Support Vector Machine
ROC	Receiver Operating Characteristic

CHAPTER 1

INTRODUCTION

1.1 Motivation

Face localization and verification is the task of detecting and verifying the identity of a person using facial features extracted from an image. Numerous methods have been proposed over the last few decades, but it still remains a difficult research problem due to the wide range of possible head poses, facial expressions, and external lighting conditions.

Recently, the availability of low-cost consumer-grade depth cameras has greatly increased due to the popularity of Microsoft's Kinect. In the near future, these devices will be embedded in our laptops and mobile phones alongside the standard color cameras. Our objective is to utilize the 3D data from a consumer depth camera to perform continuous face localization and verification. For example, when a trusted user walks up to his/her computer or picks up his/her mobile device, he/she is immediately detected, authenticated, and granted access, and when the user leaves or an intruder is detected, we can instantly revoke privileges.

There are several benefits of using a depth camera for face localization and verification. Depth cameras use infrared light to measure the 3D geometry of an environment; therefore, they are less sensitive to external illumination. In addition, with a 3D approach we can utilize a full 3D face model, which allows our technique to be more robust to changes in pose. There are also challenges associated with using a low-cost depth camera. Commodity depth cameras are often noisy and low resolution.

The goal of this thesis is to propose an accurate and real-time method for face localization and verification using a low-cost consumer-grade depth camera.

1.2 Overview

Our work consists of three major components: face detection, head pose estimation, and face verification, as depicted in Figure 1.1.

The first step is to identify the subject's face within the image. We use a standard face detector, but we significantly improve the detection process by leveraging depth information. Typically, a face detector will search an image at every location and scale to find faces. With depth data, we restrict the search to regions that are geometrically able to contain a face. As a result, we accelerate the detection process by 3.5x, and we greatly improve the accuracy by eliminating the majority of false detections.

After we identify a face, we need to determine its pose. We developed a robust, precise, and efficient model-based method for 3D head pose estimation. Our approach registers a 3D face model to the measured 3D data through a combination of the particle swarm optimization (PSO) and the iterative closest point (ICP) algorithm. We demonstrate state-of-the-art accuracy on a standard benchmark dataset.

With the face precisely localized within the depth image, we can accurately compare the facial features within the image with a database face model. The pose is used to align the reference face model to the image, and features are extracted by computing the difference between the model and the image. We leverage a learned similarity metric to determine whether or not the face in the image matches the face model. To evaluate our method, we use a combination of three datasets: two standard datasets and one we collected ourselves. We exhibit better performance than the existing 2D and 3D face verification methods on this hybrid dataset.

The remainder of this thesis is organized as follows. In Chapter 2, we describe how we improve the Viola-Jones face detector with depth information. Chapter 3 explains our approach to 3D head pose estimation. In Chapter 4, we discuss our technique for verifying the face localized within the depth image. We conclude this thesis in Chapter 5 with a summary of our contributions and closing remarks.

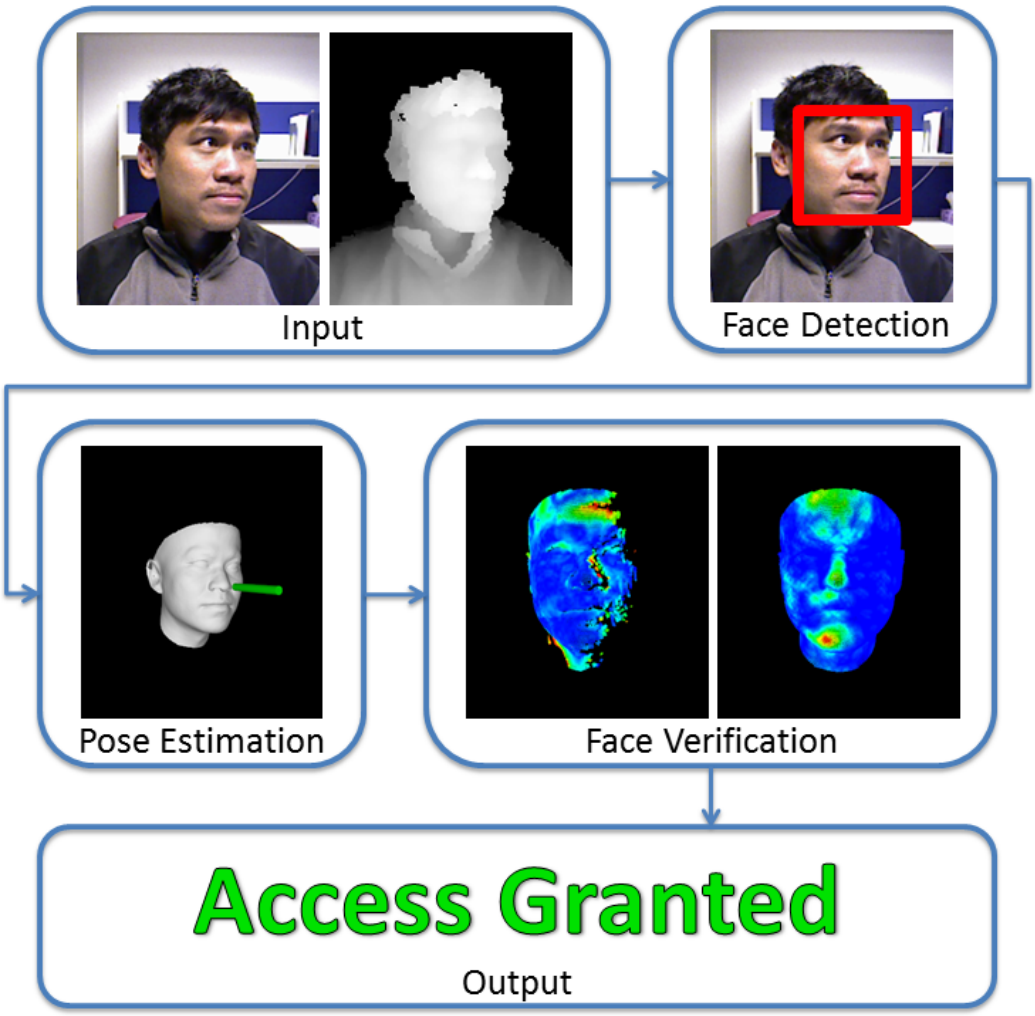


Figure 1.1: An overview of our system. Depth and color images are collected by a consumer camera. The subject's face is detected using a combination of both depth and color information. Afterwards, we estimate the pose of the user's head within the depth image, and we use the pose to align a reference model. The difference between the model and the image is computed, and the person is authenticated based on the output of our learned similarity metric.

CHAPTER 2

FACE DETECTION

2.1 Introduction

Detecting faces is an important initial step for many vision applications. Applications that typically require face detection are face analysis and biometrics, face recognition, face modeling, human-computer interaction, surveillance, etc. [1].

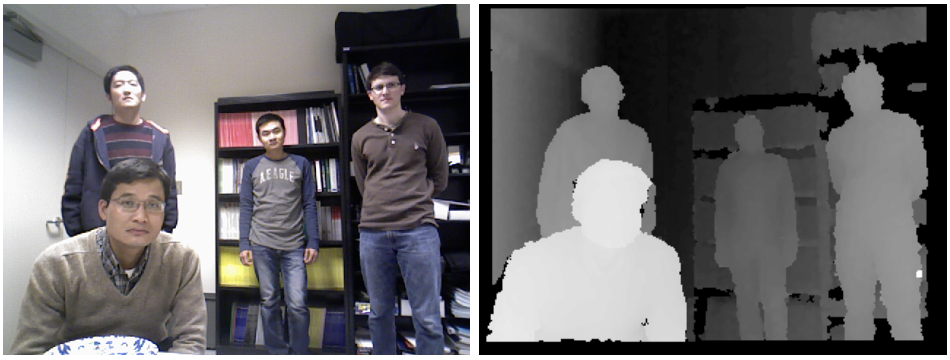
Over the past few years, the availability of color images with corresponding depth images (Figure 2.1) has increased due to the popularity of low-cost depth cameras, notably Microsoft's Kinect. The goal of this work is to utilize the additional depth data to reduce the computational cost of face detection, and in doing so, enabling new real-time applications.

Face detection methods, such as the Viola-Jones object detection framework [2, 3], identify faces by classifying sub-windows within an image as a face or non-face region. Without prior information, the size and position of a face within the image are unknown; therefore, the detector must exhaustively search the image at every position and scale.

Our approach uses depth information to identify all positions and scales within the color image that may contain a face. As a result, the face detector is no longer required to classify every sub-window within the image, which is computationally expensive and prone to false detections.

Our proposed method begins by approximating the size of a face at each pixel location based on its measured depth value. With our estimated dimensions, we analyze the geometry of the region surrounding the pixel to decide whether or not it is possible for the pixel to lie on a face. Afterwards, we construct a list of sub-windows to be classified by the Viola-Jones face detector [3].

Our technique significantly reduces the time required to detect faces, and



(a) (b)

Figure 2.1: Example (a) color and (b) depth images captured by a Kinect camera. In (b), bright pixels indicate a small depth measurement where dark pixels represent a large depth measurement.

improves accuracy by greatly reducing the number of false detections. In the following sections, we review work related to our proposed method, describe our system in detail, and present experimental results.

2.2 Related Work

Face detection is used in a wide variety of computer vision and robotic systems. Some of these systems contain sensors, such as stereo cameras, laser scanners, and depth cameras, that are capable of sensing depth. As a result, there have been a few methods proposed to accelerate face detection algorithms using depth information [4, 5].

M. Dixon *et al.* [4] describe a robotic platform where the relationship between a camera and its environment is known. Based on this knowledge, they are able to restrict the face detector's search to only geometrically able locations. For example, regions of the image that would require a person's face to be above the ceiling or below the floor are not classified by the detector. When a stereo camera or laser scanner is available, [4] further reduces their search space by eliminating sub-windows whose physical dimensions are significantly larger or smaller than the average human face.

H. Wu *et al.* [5] proposed a method to accelerate face detection using depth information computed by a pair of cameras. To reduce the cost of stereo depth estimation, [5] only computes a sparse set of depth values. Using

nearby depth samples, they estimate the size of the sub-windows within the image. Similar to [4], they avoid classifying sub-windows that are too large or too small to contain a face.

Comparable to previous work, our proposed method uses depth information to approximate the physical size of a sub-window and only classifies windows that are approximately the size of a human face. Unlike other methods, we use the geometrical information contained within the depth image to avoid classifying regions that are unlikely to contain faces.

Furthermore, techniques have been developed that utilize skin color to accelerate face detection [6]. However, skin color clustering can be affected by illumination. Depth cameras have the benefit of being robust to various lighting conditions.

2.3 Method

Our proposed method leverages depth information to identify regions within a color image that may contain a face. In addition, we use the depth data to estimate the size of the face within each region. As a result, only a small subset of windows within the image need to be classified by the face detector. Our approach avoids the exhaustive and computationally expensive search over the entire image at multiple scales.

Our technique begins by approximating the size of a face at each pixel location based on its measured depth value. Next, we analyze the geometry within the depth image to locate candidate face regions. Finally, we construct a list of sub-windows to be classified by the Viola-Jones face detector.

2.3.1 Approximating Face Dimensions

If we assume a pixel lies on a face, we can use the pixel's depth measurement to approximate the dimensions of the face. For the $(i, j)^{\text{th}}$ pixel in the image, we compute the size of the face in pixels, $s(i, j)$, using the following equation:

$$s(i, j) = \frac{f \cdot \bar{s}}{d(i, j)}, \tag{2.1}$$

(a) (b)

Figure 2.2: (a) The estimated size of the face at every pixel location, and (b) an illustration of how the size of the face changes along a scanline. As expected, the approximate face size is inversely proportionality to the depth image (Figure 2.1b).

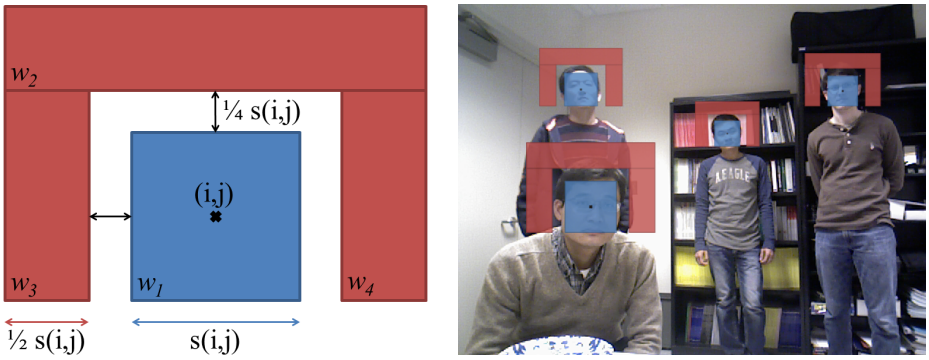
where f is the depth sensor's focal length, $d(i,j)$ is the pixel's depth value in millimeters, and $\bar{s}$ is the average width of a human face in millimeters [7]. Figure 2.2 depicts the estimated size of the face at each pixel location.

With only a pixel's depth measurement, we can avoid searching multiple scales at every pixel location, which reduces the amount of work required to detect faces, as shown in [4] and [5]. However, our proposed method goes further by analyzing a neighborhood of pixels to identify regions that are geometrically able to contain a face.

2.3.2 Identifying Candidate Face Regions

We use an adaptive template to identify candidate face regions within the image. We compare the depth measurement at the $(i,j)^{\text{th}}$ pixel to the depth values in a local neighborhood surrounding (i,j), where the size of the neighborhood is based on $s(i,j)$. We exploit the fact that depth measurements on a face should have similar value, and depth measurements left, right, and above of the face should be significantly different.

Our adaptive template is illustrated in Figure 2.3. For the $(i,j)^{\text{th}}$ pixel to lie on a face, the depth values in w_1 should be similar to $d(i,j)$, and the depth values in w_2, w_3, and w_4 should be considerably different from $d(i,j)$.



(a) (b)

Figure 2.3: (a) Adaptive template used to identify candidate face regions, and (b) an example illustrating the template at four different locations.

For each window w_k, we compute its average depth value,

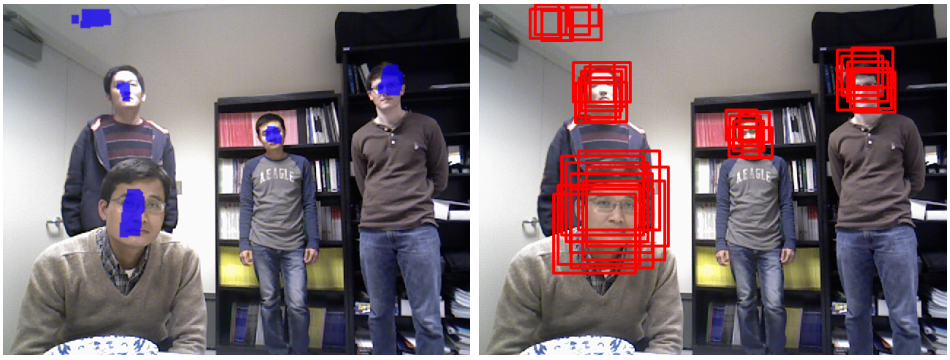
$$\mu_{w_k}(i,j) = \frac{\sum_{(u,v)\in w_k} v(u,v)\cdot d(u,v)}{\sum_{(u,v)\in w_k} v(u,v) + \epsilon}, \tag{2.2}$$

where $v(i,j)$ is a map of all valid depth measurements, and ϵ is a regularization constant to avoid division by zero. A depth value is considered valid if it is greater than zero and below a threshold T:

$$v(i,j) = \begin{cases} 1 & \text{if } 0 < d(i,j) < T \\ 0 & \text{otherwise} \end{cases}. \tag{2.3}$$

We define $T = (f \cdot \bar{s}) \,/ s^*$, where s^* is the minimum size of a face in pixels that the face detector can classify. If a pixel's depth value is larger than T, the size of the face at this location will be smaller than s^*, and the face detector will not be able to identify the face. For our implementation, $s^* = 20$ pixels and T is roughly four and a half meters.

We generate a mask, m, of candidate face regions by comparing $d(i,j)$ to



(a) (b)

Figure 2.4: (a) Mask generated by our proposed method based on the depth information in Figure 2.1b. Blue pixels in (a) indicate where $m(i,j) = 1$. (b) Sub-windows generated by our method using m and s to be classified by the face detector.

the average depth values in the surrounding windows:

$$m(i,j) = \begin{cases} 1 & \text{if } \begin{array}{l} |\mu_{w_1}(i,j) - d(i,j)| < \tau_1 \text{ and} \\ |\mu_{w_2}(i,j) - d(i,j)| > \tau_2 \text{ and} \\ |\mu_{w_3}(i,j) - d(i,j)| > \tau_3 \text{ and} \\ |\mu_{w_4}(i,j) - d(i,j)| > \tau_4 \end{array} \\ 0 & \text{otherwise} \end{cases}, \tag{2.4}$$

where $\tau_1 = 100$ mm and $\tau_2 = \tau_3 = \tau_4 = 200$ mm (these values where determined empirically). To reduce noise in the mask, we perform an open morphological operation [8]. Additionally, we expand the candidate face regions to nearby pixels with similar depth values to increase our chances of detecting faces. To generate m efficiently, we use integral images [2] to compute the window averages, $\mu_{w_k}(i,j)$, as well as to perform the morphological operations. Figure 2.4a depicts the mask of candidate face regions produced by our method.

Afterwards, we use m and s to construct a list of sub-windows to be classified by the face detector. For every pixel (i,j) where $m(i,j) = 1$, we add to the list one sub-window of size $s(i,j)$ centered on (i,j) as shown in Figure 2.4b. This list is considerably smaller than the list of all possible sub-windows within the image. As a result, our proposed method reduces not only the time it takes to detect faces, but also the number of false detections.

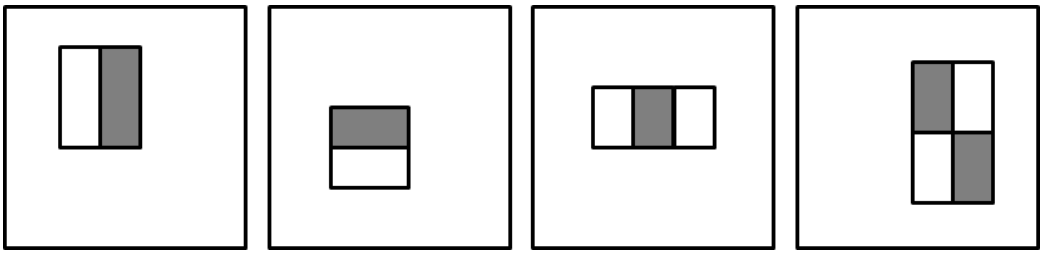


Figure 2.5: Example features used by the Viola-Jones object detection framework. The features are computed by subtracting the sum of the pixel intensities in the dark rectangle(s) from the sum of the pixel intensities in the light rectangle(s). Figure adapted from [3].

2.3.3 Classifying Sub-Windows

To classify our list of sub-windows as face or not, we require a face detector. Our approach is not reliant on a specific detector, but for our experiments we utilize the object detection framework proposed by P. Viola and M. Jones [2, 3] as it is the most commonly used face detection method.

The Viola-Jones method uses a set of Haar-like image features extracted from the sub-window, as shown in Figure 2.5, to determine whether or not a face is present. The rectangular features are beneficial because they can be computed efficiently using an integral image [2].

P. Viola and M. Jones use the AdaBoost learning algorithm [9] for selecting features and training their classifiers. The AdaBoost algorithm combines a collection of "weak learners" to form a stronger classifier. For the Viola-Jones method, each weak learner selects a single rectangle feature and threshold which best separates positive and negative examples [3]. The outputs of several weak learners are combined to determine whether or not the sub-window contains a face. Example features selected by AdaBoost are shown in Figure 2.6.

Furthermore, [3] uses a cascade of classifiers to efficiently classify a sub-window. The initial stages of the cascade contain simple classifiers that reject many negative sub-windows while retaining almost all of the positive sub-windows. The later stages contain more complex classifiers to reduce the number of false positives [3]. Figure 2.7 illustrates the cascade of classifiers used by the Viola-Jones method.

The Viola-Jones face detector was designed to rapidly identify faces within an image. However, without prior information the detector must exhaustively

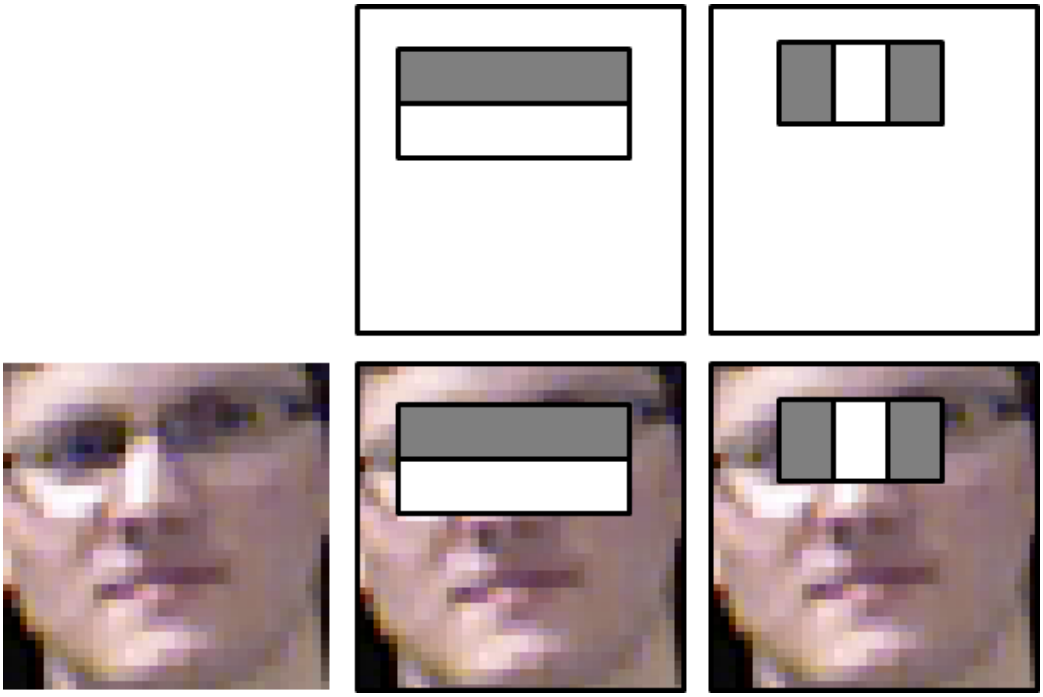


Figure 2.6: Example features selected by the AdaBoost learning algorithm. The features leverage the fact that the eyes are often darker than the upper cheeks and the bridge of the nose. Figure adapted from [3].

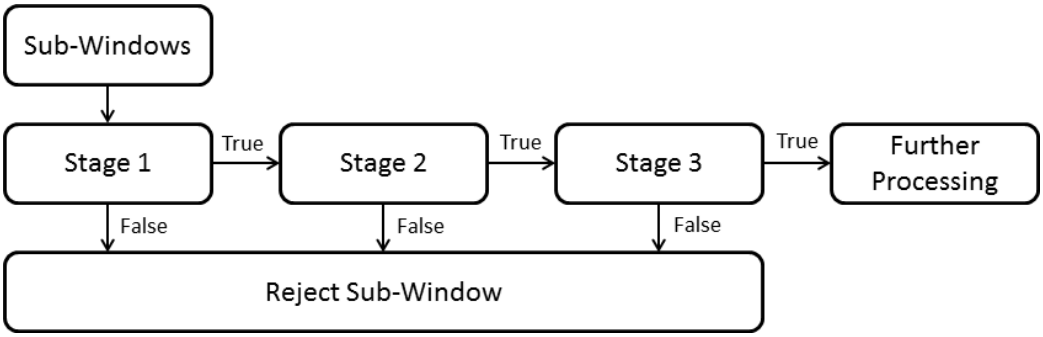


Figure 2.7: The cascade of classifiers used by the Viola-Jones face detector. The initial stages remove a large portion of the negative sub-windows with a small amount of computation. The later stages contain more complex classifiers that require more computation to obtain the desired false positive rate. Figure adapted from [3].

search the image at every position and scale to identify all the faces within the image. Our method improves the performance of the detector by restricting it to sub-windows that are likely to contain faces based on the scene's geometry.

2.4 Experimental Results

2.4.1 Data Sets

We evaluate the performance of our proposed method on the Cornell Activity Dataset (CAD-120), which features 120 videos captured by a Kinect camera [10]. The sequences contain one of four subjects performing some type of activity, such as preparing and eating food, picking up and arranging objects, cleaning, etc. Each frame consists of a color and depth image with a VGA resolution (640×480), as well as, the positions of the subject's joints within the images. Using the annotated location of the subject's head we can determine whether a detection is a true or false positive. We decided to use this dataset over other Kinect datasets [11, 12] because it contains subjects performing realistic actions in a cluttered environment. The Biwi Kinect dataset [11] and the Eurecom Kinect dataset [12] both contain a seated subject at a fixed distance from the camera with a plain background.

2.4.2 Comparison with Existing Methods

We rely on the Viola-Jones face detector to perform the final classification of the sub-windows, so the sensitivity of our method is upper-bounded by the recall of this detector. For this reason, we assume that the set of true positives detected by the Viola-Jones method contains all the faces within the CAD-120 dataset. We use this assumption to analyze the accuracy of our proposed method.

We would also like to compare our method to the techniques proposed by [4] and [5]. It is difficult to accurately contrast our methods as they have different setup requirements, and they use different 3D sensors. However, both methods use depth information in one way or another to estimate the scale at each pixel location. For comparison, we approximate [4] and [5] with our implementation by setting $m(i, j) = 1$ for every pixel (i, j).

Table 2.1: Evaluation of the Viola-Jones face detection algorithm [3] (baseline), M. Dixon *et al.* [4] and H. Wu *et al.* [5] (scale-only), and our proposed method on the CAD-120 dataset [10].

Technique	Recall	Precision	Runtime
Baseline [3]	1.00	0.57	107.31 ms
Scale-Only [4, 5]	0.95	0.73	55.10 ms
Proposed Method	0.94	0.91	30.64 ms

As shown in Table 2.1, using depth information to avoid searching multiple scales at every pixel location alone slightly improves the performance of the face detector. Dixon *et al.* [4] obtain additional speed-up by using the known calibration between the camera and its environment to avoid classifying sub-windows that would require the face to exist above the ceiling or below the floor. However, this information is not available in the CAD-120 dataset. Wu *et al.* [5] claim better performance than what is depicted in Table 2.1. It is likely they obtain their speed-up by only using a spare set of depth samples; as a result, they classify a smaller set of sub-windows. Although, it is probable this will affect the accuracy of the face detection.

Our proposed method does not require prior knowledge about the environment, nor does it sacrifice accuracy by using a subset of pixels. Our technique uses the geometrical information contained within the depth images to achieve its performance. As shown in Table 2.1, our approach accelerates face detection by 3.5x; in addition, it profoundly improves the precision without significantly impacting the recall. The total overhead incurred by our proposed method is 3.96 ms, which is included in the runtime recorded in Table 2.1. All the techniques were profiled on an Intel Core i7 CPU.

Example results from the CAD-120 dataset are shown in Figure 2.8. In order to demonstrate that our proposed method is robust to multiple subjects, we captured a few sequences with a Kinect camera, and the results are shown in Figure 2.9.

2.5 Concluding Remarks

We presented a method for improving face detection with depth. Our approach utilized the geometrical information within a depth image to identify regions within a color image that may contain a face. As a result, we avoid

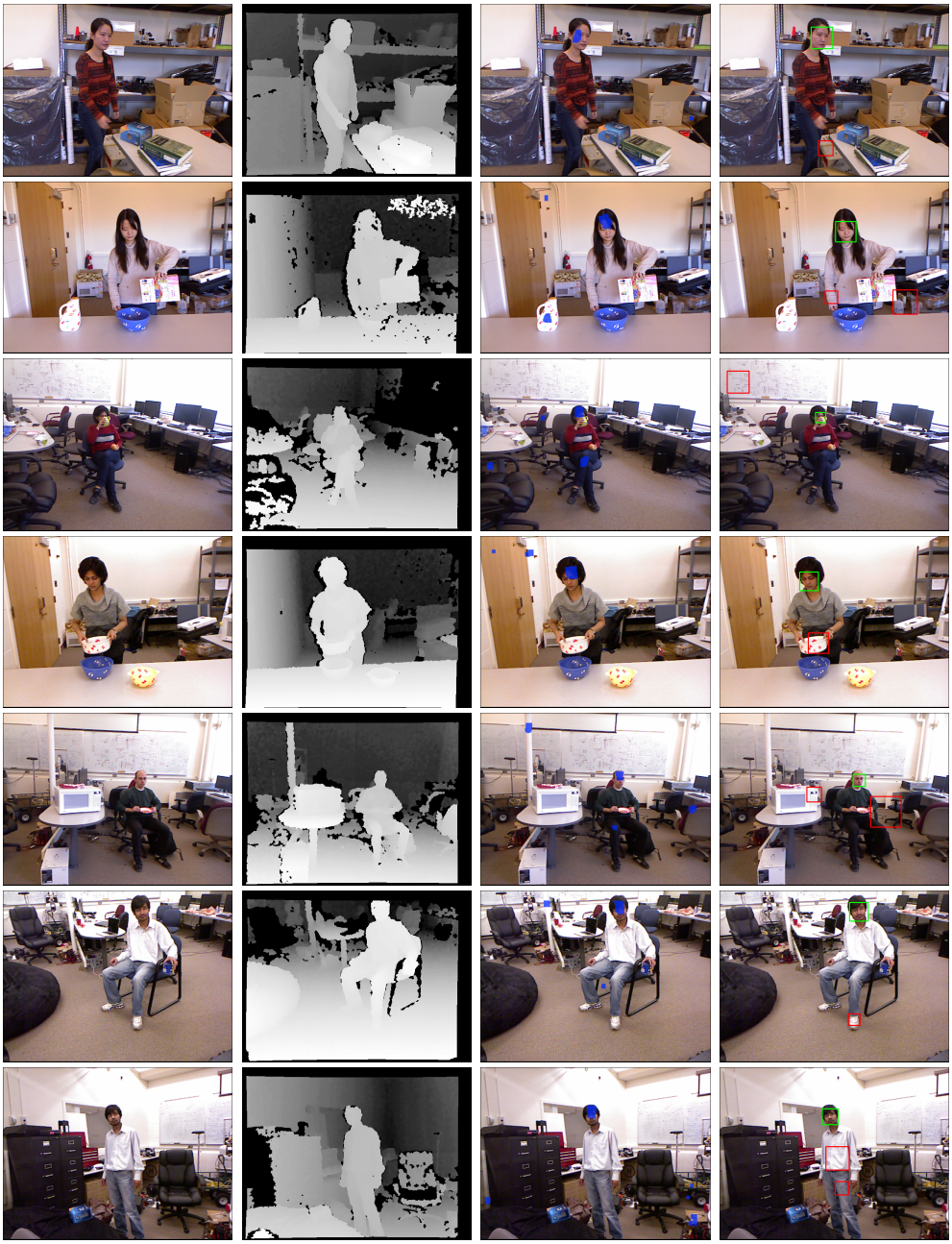


Figure 2.8: A set of RGB images (first column) and corresponding depth images (second column) from the CAD-120 dataset [10]. The mask generated by our proposed method is visualized in the third column. In the last column, faces detected by our approach are shown in green; also, false positives detected by the Viola-Jones face detector but removed by our technique are shown in red.

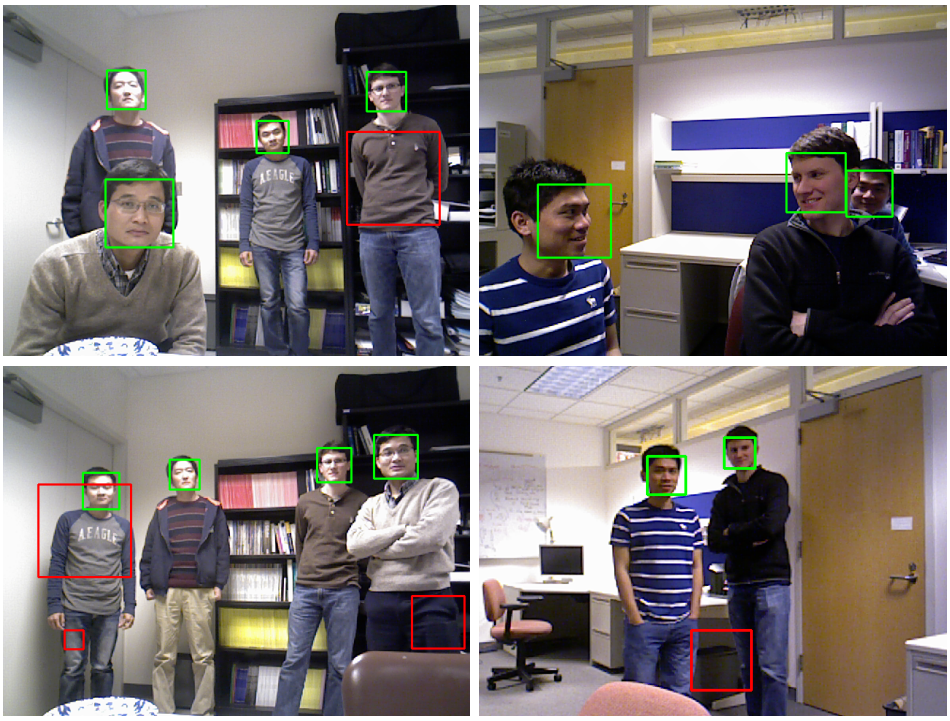


Figure 2.9: A set of results from videos captured by a Kinect that contain multiple subjects. Faces detected by our proposed method are shown in green; in addition, false detection classified by the Viola-Jones algorithm but eliminated by our approach are shown in red. Note that our approach works well even in the situation where the faces are close together.

the exhaustive and computationally expensive search over the entire image at multiple scales. Our method enables us to detect faces 3.5x faster, and it greatly reduces false detections. By providing additional post-processing time, our technique enables new real-time applications.

Our proposed method is integrated into the OpenCV library [13] through its mask generator feature, which allows us to specify the sub-windows that should be classified without modifying the face detector itself. Consequently, it is trivial to add our method to an existing system that utilizes a Kinect camera and OpenCV.

CHAPTER 3

HEAD POSE ESTIMATION

3.1 Introduction

Estimating the 3D pose of the head (rotation and position) is an important problem with applications in facial motion capture, human-computer interaction and video conferencing. It is a pre-requisite to gaze tracking, face recognition, and facial expression analysis. Head pose estimation has traditionally been performed on RGB images with rotation-specific classifiers or facial features [14], or by registering images to 3D templates [15, 16]. However, RGB-based head pose estimation is difficult when illumination variations, shadows, and occlusions are present. With the emergence of inexpensive commodity depth cameras, promising 3D techniques for body [17], hand [18], and head [11] pose estimation have been proposed.

We present an algorithm for accurate 3D head pose estimation for data acquired with commodity depth cameras. Our approach uses only 3D information, no manual intervention or training, and generalizes well to different 3D sensors. On the benchmark Biwi Kinect dataset [11], we achieve average angular errors of 2.0°, 2.1° and 2.3° for yaw, pitch, and roll, respectively, and an average translational error of 5.1 mm, while running at near real-time on a graphics processing unit (GPU). To our knowledge, this is the best accuracy reported on this dataset up to now.

We achieve this high accuracy by combining a number of concepts together in an effective manner. We register a morphable face model [19] to the measured facial data through a combination of particle swarm optimization (PSO) and the iterative closest point (ICP) algorithm. We demonstrate that together PSO and ICP simultaneously improve robustness, accuracy, and computational efficiency. Instead of creating a person-specific model during an initialization phase, we continuously adapt a morphable model to fit the

subject's face on the fly. Additionally, we dynamically weight the vertices of the morphable model to give more importance to the useful visible parts of the face, and thus handle extreme poses and partial occlusions effectively.

3.2 Related Work

Notable techniques for 3D head pose estimation employ features, pose-specific classifiers, or registration to reference 3D head models.

Sun and Yin [20] locate facial features using curvature properties to infer the head pose to within 5°. Breitenstein *et al.* [21] use the orientation of the nose as an initial estimate for head pose and refine it by comparing against pre-rendered depth images of an average face model in various poses 6° apart. Papazov *et al.* [22] introduce a triangular surface patch (TSP) descriptor to match facial point clouds to a gallery of synthetic faces and to infer their pose. Feature-based techniques fail when the features cannot be detected, *e.g.* in the case of extreme rotations or partial occlusions.

Among the classifier-based techniques is the work of Seemann *et al.* [23], where they detect faces in RGB images and estimate the head pose from the disparity map of a stereo camera using a neural network for each rotation. Fanelli *et al.* [11] train random classification and regression forests with range image patches for head detection and pose estimation. Their technique achieves good accuracy on high and low quality depth data [24, 11]. Tulyakov *et al.* [25] use cascaded tree classifiers and achieve higher accuracies than Fanelli *et al.* Classifier-based techniques require extensive training with large datasets. Moreover, classifiers trained on one 3D sensor do not generalize well to others.

An alternate approach registers a 3D head model to the measured data using the rigid/non-rigid ICP algorithm. Previous promising methods, *e.g.* [26, 27, 28, 29], employ 3D deformable model fitting to create person-specific models for head pose estimation. However, these existing methods require offline initialization with significant cooperation from the user to construct the subject-specific reference models. In contrast, we refine the morphable model's shape continuously to fit the subject while simultaneously estimating the head pose.

To ensure robustness to facial expressions, a number of the existing de-

formable model fitting based approaches, *e.g.* [30, 27], include only the less deformable eyes and nose regions of the face in the reference model. However, when the parts of the face that are included in the reference model are not visible, *e.g.* when the head is tilted back, such that the eyes and nose regions are not visible, the reference model matches poorly to the observed data, resulting in inaccurate pose estimation. In order to address this, we instead employ the entire morphable model and dynamically weight the regions of the model based on which parts of the face are visible.

Techniques for facial animation capture with high [31, 32, 33] and low [30, 34] quality 3D scans also employ very precise morphable model fitting. These techniques require significant manual interaction to create very detailed person-specific models. Also, these studies do not directly report the accuracy of head pose estimation, but presumably perform sufficiently well to enable effective facial expression capture.

To avoid deformable model fitting, some methods directly use facial data from the 3D video sequence as a reference. For example, Padeleris *et al.* [35] use the first frame, Bar *et al.* [36] use frontal, left, and right profile faces, and Martin *et al.* [37] employ 100 frames with faces in different poses. When the absolute pose of the reference face is unknown, these techniques merely provide the pose of the face relative to the reference and not the absolute head pose. Furthermore, the quality of these references is low, as they often contain holes and noise. Generally, including more views of the face in the reference model tends to improve accuracy. Tulyakov *et al.* [25] achieve the best accuracy among these approaches by registering multiple frames and volumetrically averaging them to produce a higher quality reference model.

To register the reference model to the measured data, ICP and its variants are often used. However, ICP fails to converge to the correct solution when it is initialized poorly. To overcome this, Padeleris *et al.* [35] employ the stochastic PSO algorithm [38] to register facial surfaces. However, PSO also suffers from slow and/or premature convergence to a local optimum. Recently, Qian *et al.* [39] proposed an optimization algorithm that combines PSO and ICP to overcome their individual limitations and successfully applied it to estimate the 26-dimensional pose of 3D hands. Their work has inspired us to employ a combination of PSO and ICP to accurately estimate the 3D head pose. As an extension of their work, we provide a detailed analysis to understand the underpinnings and conditions for success of combining

PSO and ICP for 3D surface registration.

3.3 Method

3.3.1 Reference Model

For the reference model, we utilize the 3D Basel Face Model [40], which is a 3D morphable model [19]. With a morphable model, a facial surface, comprised of a set of 3D vertices $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N)$, can be represented as a linear combination of an average face shape $\boldsymbol{\mu}$ and a set of face shape components $\boldsymbol{S} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_M)$:

$$\mathbf{V} = \boldsymbol{\mu} + \sum_{m=1}^{M} \alpha_m \boldsymbol{s}_m = \boldsymbol{\mu} + \boldsymbol{S}\boldsymbol{\alpha}. \tag{3.1}$$

Figure 3.1 illustrates the 3D morphable face model. Parts of the observed face may not match the reference model (*e.g.*, due to facial hair, or eyewear); therefore, we use a weight vector, $\mathbf{W} = (w_1, w_2, ..., w_N)$, to represent the confidence of each vertex in the reference model. For the initial frame, we set our shape vector $\mathbf{V}_0$ to the average face $\boldsymbol{\mu}$ and our weight vector $\mathbf{W}_0$ to unity for all vertices. In subsequence frames, we update the shape and weight vector to better match the observed face.

3.3.2 Cost Function

The pose of the head is indicated by a 6-dimensional vector $\mathbf{x} = (\theta_x, \theta_y, \theta_z, t_x, t_y, t_z)$, where θ_i and t_i represent a rotation about and a translation along the axis i. We evaluate a hypothetical pose $\mathbf{x}$ for an observed depth image d_o by first rendering a depth image $d_\mathbf{x}$ and a weight image $w_\mathbf{x}$ of the reference model in the pose $\mathbf{x}$:

$$(d_\mathbf{x}, w_\mathbf{x}) = \text{Render}\left(\mathbf{x}, \mathbf{V}_k, \mathbf{W}_k, \mathbf{K}\right), \tag{3.2}$$

where $\mathbf{V}_k$ and $\mathbf{W}_k$ are the current shape and weight of the reference model, and $\mathbf{K}$ is the camera's intrinsic calibration matrix. Figure 3.2 depicts an

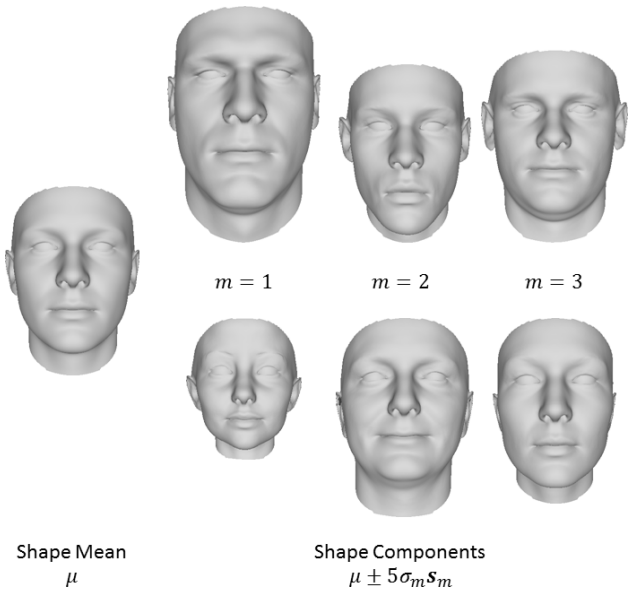


Figure 3.1: The mean shape and the shape vectors generated by adding or subtracting one of the first three shape components of the 3D Basel Face Model. Figure adapted from [40].

example of an observed depth image, as well as a rendered depth and weight image. Each depth pixel in d_o and $d_{\mathbf{x}}$ has a corresponding 3D vertex generated by back-projection:

$$\mathbf{v}_o(i,j) = \mathbf{K}^{-1} d_o(i,j) \begin{bmatrix} i \\ j \\ 1 \end{bmatrix}, \quad \mathbf{v}_{\mathbf{x}}(i,j) = \mathbf{K}^{-1} d_{\mathbf{x}}(i,j) \begin{bmatrix} i \\ j \\ 1 \end{bmatrix}. \tag{3.3}$$

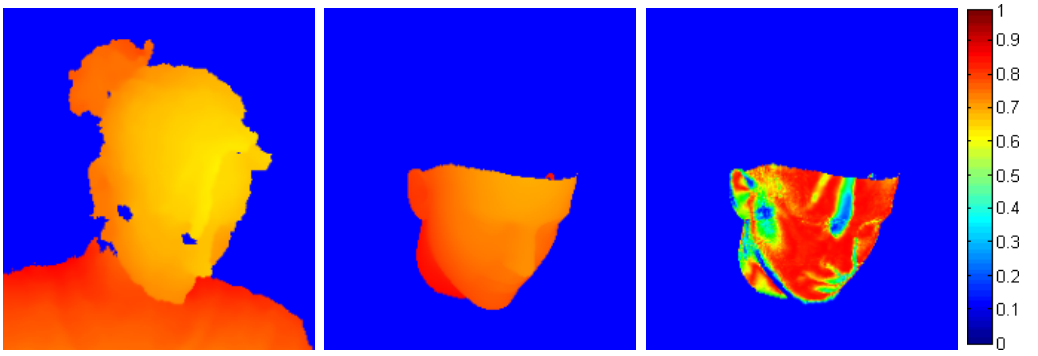
In addition, we compute a normal vector $\mathbf{n}_{\mathbf{x}}$ for each vertex in $\mathbf{v}_{\mathbf{x}}$ using the relative position of its neighboring vertices:

$$\mathbf{n}_{\mathbf{x}}(i,j) = \left[\mathbf{v}_{\mathbf{x}}(i+1,j) - \mathbf{v}_{\mathbf{x}}(i,j)\right] \times \left[\mathbf{v}_{\mathbf{x}}(i,j+1) - \mathbf{v}_{\mathbf{x}}(i,j)\right]. \tag{3.4}$$

To factor out the effect of outliers, which are commonly observed with low-cost depth cameras, we generate a subset, $\mathcal{R}$, of reliable vertices to be compared:

$$\mathcal{R} = \left\{(i,j) \middle| \, \|\mathbf{v}_o(i,j) - \mathbf{v}_{\mathbf{x}}(i,j)\| < \tau, (i,j) \in \mathcal{O} \cap \mathcal{H}\right\}, \tag{3.5}$$

where $\mathcal{O}$ and $\mathcal{H}$ are the sets of valid (non-zero) pixels in the observed and hypothetical depth images, respectively. In our experiments, we empirically set $\tau = 3$ cm.



(a) (b) (c)

Figure 3.2: To evaluate a pose, we compare the (a) observed depth image with the (b) rendered depth image weighted by the (c) rendered weight image.

We then compute the following cost function to quantify the discrepancy between the observed and the hypothetical data:

$$E(\mathbf{x}) = E_v(\mathbf{x}) + \lambda E_c(\mathbf{x}), \tag{3.6}$$

where

$$E_v(\mathbf{x}) = \frac{\sum_{(i,j)\in\mathcal{R}} w_{\mathbf{x}}(i,j) \left[\left(\mathbf{v}_o(i,j) - \mathbf{v}_{\mathbf{x}}(i,j)\right)^T \mathbf{n}_{\mathbf{x}}(i,j) \right]^2}{\sum_{(i,j)\in\mathcal{R}} w_{\mathbf{x}}(i,j)} \tag{3.7}$$

and

$$E_c(\mathbf{x}) = \left[1 - \sum_{(i,j)\in\mathcal{R}} w_{\mathbf{x}}(i,j) \Big/ \sum_{(i,j)\in\mathcal{H}} w_{\mathbf{x}}(i,j) \right]^2 . \tag{3.8}$$

The term $E_v(\mathbf{x})$ measures the point-to-plane distance between corresponding vertices on the two surfaces, whereas $E_c(\mathbf{x})$ measures the extent to which the depth images coincide with each other (*i.e.*, it penalizes the hypothetical and observed depth images for not overlapping). The parameter λ designates the relative importance of the two terms, and it was empirically set to 350.

3.3.3 Optimization

In order to compute the pose, we employ a combination of particle swarm optimization (PSO) and the iterative closest point (ICP) algorithms.

PSO [41] uses a set of particles, that evolve through social interactions over a series of generations, to search for a global optimum in a non-convex parameter space. For head pose estimation, each particle represents a head pose $\mathbf{x}$ and has a corresponding cost, $E(\mathbf{x})$, specified by Eq. (3.6). Each particle keeps track of the position $\mathbf{x}^*$ where it has observed the lowest cost, $E(\mathbf{x}^*)$, across all generations. The best position across all particles and generations is indicated by $\mathbf{x}_g^*$. At generation t, every particle stochastically updates its position $\mathbf{x}$ and velocity $\mathbf{u}$ based on its position relative to $\mathbf{x}^*$ and $\mathbf{x}_g^*$ [42]:

$$\begin{aligned} \mathbf{u}_{t+1} &= \gamma \left(\mathbf{u}_t + \alpha \xi_1 \left(\mathbf{x}^* - \mathbf{x}_t \right) + \beta \xi_2 \left(\mathbf{x}_g^* - \mathbf{x}_t \right) \right) \\ \mathbf{x}_{t+1} &= \mathbf{x}_t + \mathbf{u}_{t+1}, \end{aligned} \tag{3.9}$$

where the constants α, β, and γ are the cognitive, social, and constriction factors, respectively, and ξ_1 and ξ_2 are uniform random variables $\in [0, 1]$. Based on [42], we set $\alpha = \beta = 2.05$ and $\gamma = 0.7298$.

During the first generation ($t = 0$), the particles' positions are initialized randomly, and their velocities are set to zero. For the initial frame, the particles' positions are generated by randomly sampling a normal distribution with the mean set to the frontal pose. For subsequent frames, half of the particles are initialized in this way, and the other half use a normal distribution with a mean set to the previous frame's pose estimate.

To prevent unlikely head poses, we bound the parameter space: $\theta_x \in [-60°, 60°]$ for pitch, $\theta_y \in [-90°, 90°]$ for yaw, and $\theta_z \in [-45°, 45°]$ for roll. For translation, we force the centroid of the reference model to remain within a certain distance ($\sim$ 10 cm) from the location we identified during face detection.

For each particle and for each generation, we perform a few iterations of ICP [43] to push particles toward a local optimum of the parameter space. Instead of minimizing $E(\mathbf{x})$ directly, which would be difficult because of the implicit rendering of $d_{\mathbf{x}}$ and $w_{\mathbf{x}}$, we solve a simpler problem. We fix $d_{\mathbf{x}}$ and $w_{\mathbf{x}}$, and we solve for a rigid body transformation that pushes $\mathbf{v}_o$ toward $\mathbf{v}_{\mathbf{x}}$. Afterwards, we can update the particle's pose by applying the inverse transformation to it.

A rigid body transformation is defined as a rotation matrix,

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

and a translation vector $\boldsymbol{t} = [t_x\, t_y\, t_z]^T$. We solve for the parameters of $\boldsymbol{R}$ and $\boldsymbol{t}$ by minimizing the point-to-plane distance between $\mathbf{v}_o$ and $\mathbf{v}_\mathbf{x}$,

$$\arg\min_{\boldsymbol{R},\boldsymbol{t}} \sum_{(i,j)\in\mathcal{R}} w_\mathbf{x}(i,j) \left[\left(\boldsymbol{R}\mathbf{v}_o(i,j) + \boldsymbol{t} - \mathbf{v}_\mathbf{x}(i,j)\right)^T \mathbf{n}_\mathbf{x}(i,j) \right]^2, \tag{3.11}$$

which is the same distance measured by the first term of the cost function, E_v. To obtain a closed-form solution we utilize the small angle approximation [44]. After solving for the rigid body transformation, we modify $\mathbf{v}_o$ by applying the transformation and re-projecting each vertex. We repeat this process for a fixed number of iterations. Upon completion, we apply the inverse rigid body transformation to update the particle's position $\mathbf{x}$.

As a trade-off between accuracy and computation time for our combined PSO and ICP optimization procedure, we used a set of 10 particles and 5 generations for PSO, and 3 iterations of ICP. After the optimization terminates, we provide the position of the best particle over all the generations, $\mathbf{x}_g^*$, as the final pose estimate for the face in the current frame.

3.3.4 Model Update

Once the head pose has been estimated, we update the shape and weights of the reference model to match the observed face in the current frame. Utilizing the estimated pose, we identify point correspondences between the reference face model and the observed data by transforming and projecting the vertices of the model into the observed vertex map $\mathbf{v}_o$,

$$\begin{bmatrix} i & j & 1 \end{bmatrix}^T = \mathbf{K}\left(\boldsymbol{R}\mathbf{v}_n + \boldsymbol{t}\right), \quad \hat{\mathbf{v}}_n = \boldsymbol{R}^{-1}\left(\mathbf{v}_o(i,j) - \boldsymbol{t}\right), \tag{3.12}$$

where $\boldsymbol{R}$ and $\boldsymbol{t}$ are the rotation matrix and translation vector parameterized by the estimated pose $\mathbf{x}_g^*$, $\mathbf{v}_n$ is the n-th element in the shape vector $\mathbf{V}_k$, and $\hat{\mathbf{v}}_n$ is a vertex in $\mathbf{v}_o$ which is the point corresponding to vertex $\mathbf{v}_n$. In

addition, we define a mask:

$$m_n = \begin{cases} 1 & \text{if } \|\mathbf{v}_n - \hat{\mathbf{v}}_n\| < \delta \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

for rejecting corresponding points that are too far apart ($\delta = 1$ cm). We solve for the coefficients of the morphable model that minimize the following linear system of equations:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \left\| \mathbf{M} \left(\boldsymbol{\mu} + \boldsymbol{S}\boldsymbol{\alpha} - \hat{\mathbf{V}} \right) \right\|^2 + \nu \|\boldsymbol{\alpha}\|^2, \tag{3.14}$$

where $\hat{\mathbf{V}} = (\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, ..., \hat{\mathbf{v}}_N)$, $\mathbf{M} = \text{diag}(m_1, m_2, ..., m_N)$, and ν is a large constant to prevent unrealistic face shapes. Afterwards, we update the shape of the reference model,

$$\mathbf{V}_{k+1} = \eta \left(\boldsymbol{\mu} + \boldsymbol{S}\boldsymbol{\alpha}^* \right) + (1 - \eta) \mathbf{V}_k, \tag{3.15}$$

where $\eta = 0.1$ is a damping parameter introduced to prevent the shape from drastically changing between frames. In addition, we update the weights of the reference model as:

$$w_n = \exp\left(-\|\mathbf{v}_n - \hat{\mathbf{v}}_n\|^2 / \sigma_w \right), \tag{3.16}$$

where w_n and $\mathbf{v}_n$ are the n-th elements in the weight vector $\mathbf{W}_{k+1}$ and the shape vector $\mathbf{V}_{k+1}$, respectively, and $\sigma_w = 0.01$.

3.3.5 Dynamic Swarm

Through our experimentation we observed that the entire swarm of particles is not always necessary to accurately estimate the pose in every frame. If we correctly estimate the pose in the previous frame, then only a few particles are required to update the pose in the subsequent frame. For this reason, we propose a variant of our approach that dynamically updates the number of particles in the swarm. By dynamically resizing the swarm, we can improve the runtime performance of the algorithm without significantly affecting accuracy.

The distance term E_v of the cost function (Eq. 3.6) gives us an indication of the accuracy of the estimated head pose, $\mathbf{x}_g^*$. When we correctly identify the pose, the value of $E_v(\mathbf{x}_g^*)$ should be the expected measurement error of the depth camera. Therefore, we can update the size of the swarm for the next frame using the following procedure:

$$P_{k+1} = \begin{cases} P_{max} & \text{if } E_v(\mathbf{x}_g^*) \gg e \\ P_k + 1 & \text{else if } E_v(\mathbf{x}_g^*) > e \text{ and } P_k < P_{max} \\ P_k - 1 & \text{else if } E_v(\mathbf{x}_g^*) \leq e \text{ and } P_k > P_{min} \\ P_k & \text{otherwise} \end{cases} \tag{3.17}$$

where P_k is the number of particles in the current frame k, $P_{min} = 1$ and $P_{max} = 10$ are the minimum and maximum number of particles in the swarm, respectively, and e is the expected measurement error of our sensor. For the Kinect camera, we experimentally determined $e \approx 5$ mm.

3.4 Experimental Results

3.4.1 Data Sets

We measured the performance of our method and compared it with state-of-the-art algorithms on two datasets. The Biwi Kinect Head Pose dataset, acquired with a Kinect sensor, contains over 15K RGB and depth images of 20 subjects recorded in 24 sessions [45]. It has large head rotations, long hair, and occlusions. For each frame, a ground truth binary mask of the face pixels, as well as the 3D orientation of the head and the location of its center, are provided. On this dataset, we first locate the head using the method described in Chapter 2 and then estimate its pose.

The ETH Face Pose Range Image dataset by Breitenstein *et al.* contains 10K range images of 20 people [21]. These data are of higher quality than the Biwi Kinect data, and were acquired with a stereo enhanced structured light sensor [46]. In the ETH dataset, depth data is only available for the head region, thus we did not apply head detection on it.

3.4.2 Evaluation of Our Approach

Table 3.1 shows the average absolute errors for the yaw, pitch, and roll angles achieved by our algorithm on the Biwi Kinect dataset. It also contains the average positional errors for the head center and the accuracy of pose estimation. Accuracy is defined as the percentage of frames with an $L2$ norm of angular errors less than 10°.

On the Biwi Kinect dataset, we achieved angular errors of 2.0°, 2.1° and 2.3° for the yaw, pitch, and roll, a translational error of 5.1 mm and an accuracy of 95.0% with our proposed algorithm (row 1 in Table 3.1). In order to understand the contribution of each of the individual concepts employed in our algorithm, we additionally evaluated its performance with different configurations (Table 3.1).

Observe that between the morphable model (first row in Table 3.1) and the average face model (second row in Table 3.1), the morphable model consistently performed slightly better by allowing a better fit to each specific subject. Although, we did not fit the morphable model very precisely to the observed face. It seems that personalization of the face model is important for head pose estimation, but further investigation is required to establish this conclusively.

Next, we investigated the effect of combining PSO and ICP for optimization. Keeping all other parameters constant, we performed the optimization with 40 iterations of ICP only. In addition, we used PSO only with 25 particles and 40 generations. This configuration was previously shown to be effective [35]. Individually, PSO (fourth row in Table 3.1) performed the worst, considerably worse than ICP (third row in Table 3.1), plausibly because of PSO's slow convergence rate and susceptibility to premature convergence. ICP, by itself, performed better than PSO, but tended to fail for large angles of rotation. Lastly, analogous to what Qian *et al.* [39] observed for 3D hand pose estimation, we found that the combined PSO and ICP optimization method produced the most accurate results for 3D head pose estimation, as well.

The fifth row of Table 3.1 lists the performance of our algorithm for the case when we employed only the distance term E_v (Eq. 3.6) to measure the similarity between two 3D point clouds. On comparing these results with those in the first row of Table 3.1, where we used both the error terms in Eq.

(3.6), it can be concluded that the 2D overlap term (E_c) helps to improve accuracy of head pose estimation considerably. Breintenstien *et al.* [21] made similar observations in their work where they found the overlap term to positively impact the accuracy of head pose estimation.

The effect of dynamically re-weighting parts of the reference model to best match the instantaneous appearance of the observed 3D face can be evaluated by comparing the first and sixth rows of Table 3.1. To obtain the values listed in the sixth row, we set all the weights of the reference model to unity and kept them constant over time instead of dynamically varying them. Dynamically re-weighting the reference model improves accuracy and helps to robustly handle partial occlusions of the face (*e.g.*, the first and sixth rows of Figure 3.3). Recently, Tulyakov *et al.* [25] also achieved a good pose estimation result by adopting a slightly different dynamic re-weighting scheme.

Finally, the result of dynamically changing the number of particles in the swarm can be seen by comparing the first and the seventh rows of Table 3.1. The accuracy is reduced by a small amount, but the runtime performance is greatly improved. Our implementation of the proposed method with a fixed number of particles runs in approximately 160 ms, where the variant of the proposed method with a dynamic number of particles runs in about 60 ms on an NVIDIA GeForce GTX 660 GPU. With a dynamic swarm we reduce the average runtime by 100 ms.

3.4.3 Comparison with Existing Methods

A number of recent algorithms [21, 11, 26, 35, 27, 37, 25, 22] for 3D head pose estimation have also been evaluated on the benchmark Biwi Kinect [11] and ETH [46] datasets. For all these methods, except for Padeleris *et al.*'s [35], we list the results reported by the authors in Tables 3.1 and 3.2, along with a summary of their methods. Padeleris *et al.* report average errors for only 91.4% of the frames on which their algorithm succeeded. Furthermore, when their algorithm failed, they re-initialized head pose tracking with the ground truth pose (personal communications with the authors). For a fair comparison, we re-implemented their PSO-based algorithm and reported its results for the entire Biwi Kinect dataset in Table 3.1.

Table 3.1: The average absolute angular errors (in degrees) and the average translational error (in mm) on the Biwi Kinect dataset, for different configurations of our proposed algorithm, and for other existing state-of-the-art head pose estimation algorithms [11, 25, 35, 37, 27, 26, 22].

Method	**Model**	**Weights**	**Swarm**	**Optim.**	**Cost function**	**Errors**				**Accuracy**
						Yaw [°]	Pitch [°]	Roll [°]	Location [mm]	
Proposed*	Morph	Dyn	Fix	PSO+ICP	All terms	2.0	2.1	2.3	5.1	95.0%
Proposed	Average	Dyn	Fix	PSO+ICP	All terms	2.2	2.2	2.6	6.2	92.3%
Proposed	Morph	Dyn	Fix	ICP	Distance	3.7	4.3	4.0	11.1	84.2%
Proposed	Morph	Dyn	Fix	PSO	All terms	7.7	8.4	6.0	23.3	73.7%
Proposed	Morph	Dyn	Fix	PSO+ICP	Distance	4.4	3.4	4.0	9.0	87.1%
Proposed	Morph	Fix	Fix	PSO+ICP	All terms	2.7	3.1	3.2	5.9	89.7%
Proposed	Morph	Dyn	Dyn	PSO+ICP	All terms	2.3	2.3	2.6	5.4	93.6%
Fanelli [11]	N/A	N/A	N/A	RF	N/A	8.9	8.5	7.9	14.6	79.0%
Tulyakov [25]	N/A	N/A	N/A	CT	N/A	4.7	5.3	-	-	-
Padeleris [35]	Frame 0	N/A	N/A	PSO	Distance	11.1	6.6	6.7	13.8	76.0%
Martin [37]	100 Frames	N/A	N/A	ICP	Distance	2.6	2.5	3.6	5.8	85.0%
Rekik [27]	Morph	N/A	N/A	PF	Color + Distance	5.1	4.3	5.2	5.1	-
Baltrusaitis [26]	CLM-Z	N/A	N/A	RLMS	Color + Distance	6.3	5.1	11.3	-	-
Papazov [22]	TSP	N/A	N/A	N/A	TSP match	3.0	2.5	3.8	8.4	-

Table 3.2: The average absolute angular errors for yaw and pitch, for the methods by Breitenstein *et al.* [21], Fanelli *et al.* [24], and for our method, on the ETH database [24]. No head localization was performed.

Method	**Error**		**Accuracy**
	Yaw [°]	Pitch [°]	
Breitenstein [21]	6.1	4.2	80.8%
Fanelli [24]	5.7	5.1	90.4%
Proposed	2.9	2.3	98.9%

On the Biwi Kinect dataset (Table 3.1), our proposed algorithm produced the lowest rotational errors, which were lower than those of the existing classifier-based [11, 25], rigid model-fitting-based [35, 37], non-rigid model-fitting-based [26, 27] and surface patch descriptor-based [22] approaches for head pose estimation. Despite using only the 3D information, our algorithm performed even better than the algorithms that employed both depth and color information [27, 26, 22].

We observed similar results on the ETH dataset (Table 3.2), where our algorithm outperformed the existing methods at estimating the yaw and pitch rotations (this dataset does not contain rotations about the roll axis). Example results are shown in Figure 3.4.

For all frames of the Biwi Kinect dataset, our algorithm also resulted in the smallest translational error (5.1 mm) of all the purely 3D-based head pose estimation algorithms (Table 3.1). Rekik *et al.*'s algorithm produced the same translational error as ours, but they used both color and depth data.

We also applied our head pose estimation algorithm to data acquired with a SoftKinetic DS325 time-of-flight camera. We observed similar accuracy for head pose estimation with this sensor as well (Figure 3.5).

To further understand the effect of noise and resolution, we re-projected faces in the Biwi Kinect dataset, which are originally ~ 1 m away from the sensor, to 1.5 m, 2 m and 2.5 m and added progressively increasing depth-dependent noise using the model proposed by Khoshelham and Elberink [47]. With increasing distance from the sensor our head pose estimation technique resulted in a linear increase in the yaw, pitch, and roll errors at a rate of 1.25°/m and a linear decrease in accuracy at a rate of 6.7%/m.

Figure 3.3: A set of RGB images (first column) and the corresponding depth images (second column) from the Biwi Kinect dataset [45]. The last column shows the head pose estimated by our method along with the dynamic weights assigned to different parts of the reference model.

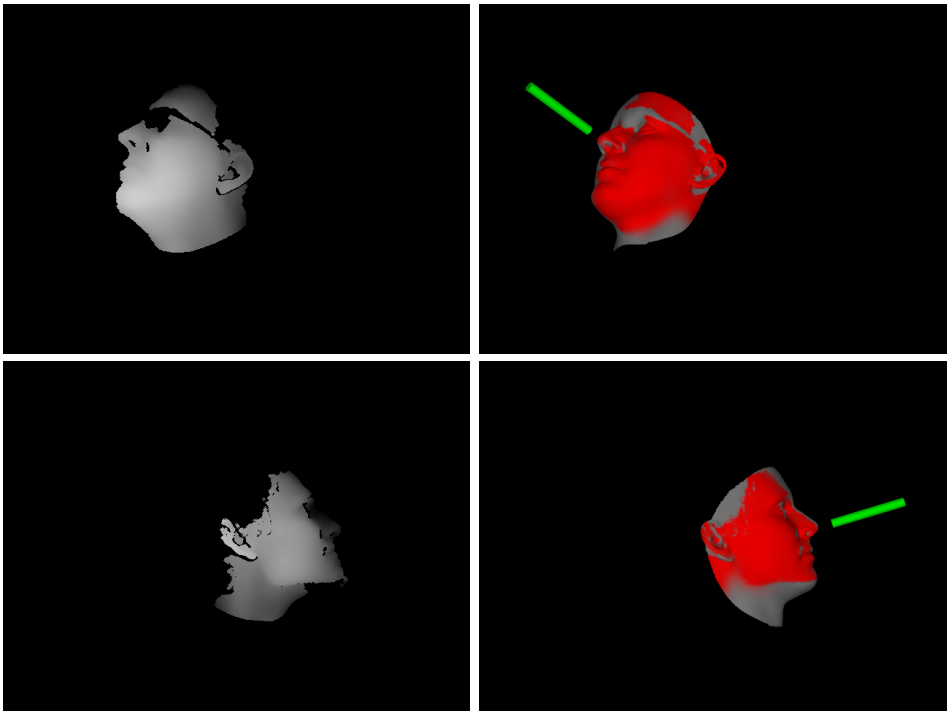


Figure 3.4: Range images from the ETH Face Pose Range Image dataset [21] are displayed in the left column. The head poses estimated by our approach are shown in the right column.

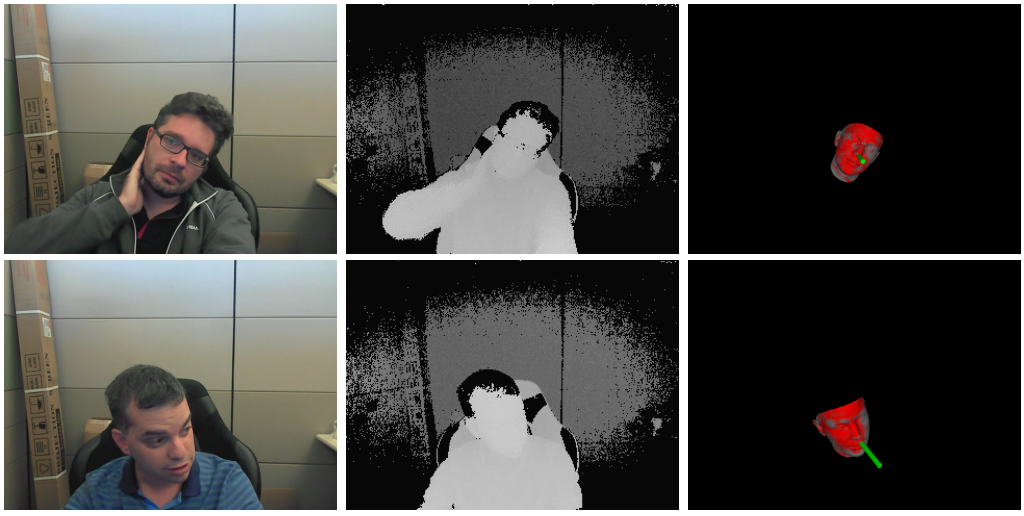


Figure 3.5: A set of color (first column) and depth images (second column) acquired by the SoftKinetic DS325 camera. The head poses estimated by our algorithm are depicted in the rightmost column (colored according to the adaptive weights).

3.4.4 Combined PSO and ICP Optimization

To better understand the peculiarities of the combined PSO and ICP algorithm, with different cost functions E and E_v, we considered the problem of registering (*i.e.* finding the optimal translation t_x) a 1D curve (in blue in Figure 3.6a) with a set of sampled noisy points (in red in Figure 3.6a). The first term of our cost function, $E_v(t_x)$, contains many local minima (Figure 3.6b). Consequently ICP, which only optimizes $E_v(t_x)$, quickly converges toward one of these, depending upon the initialization. The trajectories for two different initializations are shown in red in Figure 3.6b.

The overlap term $E_c(t_x)$ in our cost function, on the other hand, is quasi-convex (Figure 3.6c). Although $E_v + E_c$ is still non-convex, adding E_c makes the global optimum more evident and local minima less pronounced (Figure 3.6d); this explains the worse results reported in Table 3.1 (fifth row) for the use of E_v alone. Nonetheless, optimization with PSO remains problematic: particle 1 in Figure 3.6d (on the right) moves towards particle 0 and misses the global optimum, leading to premature convergence into a local minimum of $E_v + E_c$. Additionally, since PSO randomly samples the cost function without using the gradient, convergence towards the global optimum is generally slow. This explains the poor accuracy for head pose estimation that we observed when we employed PSO only (Table 3.1).

Figure 3.6e shows the trajectories of the combined PSO and ICP algorithm, where we apply ICP to each particle before the PSO update. ICP moves each particle to a local minimum of E_v (green triangle), thus making it less likely for PSO to skip over the basin of attraction of the global optimum. Note that, assuming that ICP converges, each particle is then constrained to lie in a local minimum of E_v. Since the local minima of E_v are generally only slightly offset with respect to the corresponding local minima of $E_v + E_c$, the combined optimization is generally more efficient and effective than PSO alone, as measured for our head pose estimation algorithm in Table 3.1. It is in fact sufficient for a particle to lie in the basin of attraction of the global optimum of E_v to quickly converge towards it (see the left particle in Figure 3.6e). We also noted that the basins of attraction for E_v and $E_c + E_v$ are slightly different; as a consequence, ICP may contribute to move a particle out of a local basin of attraction of $E_c + E_v$ (see the right particle in Figure 3.6f), thus potentially preventing premature convergence, increasing

the overall mobility of the particles and favoring a wider exploration of the parameter space. On the other hand, we also noted that, because of ICP, the left particle in Figure 3.6e oscillates around the same local minimum for two consecutive generations. This represents a potential drawback of any hybrid PSO algorithm, that can be mitigated by modifying the α, β and γ parameters in PSO.

3.5 Concluding Remarks

We introduced a head pose estimation method for commodity depth cameras that results in best-in-class accuracy on benchmark datasets. It requires no initialization, handles extreme rotations and partial occlusions, and efficiently registers facial surfaces. Numerous factors contribute to the success of our algorithm: the overlap term (E_c) in the cost function, the combined PSO and ICP algorithm, dynamically adapting the weights of the face model, and the adoption of a morphable face model. While these concepts have each been introduced individually in previous studies, the contribution of our work lies in combining these disparate ideas in an effective manner to significantly improve the accuracy of 3D head pose estimation. Our work also presents for the first time a systematic quantitative assessment of the contribution of each of these various factors in improving the accuracy of head pose estimation. Building upon the work of Qian *et al.* [39] we also provide deeper insights into the workings of the combined PSO and ICP optimization for 3D surface registration.

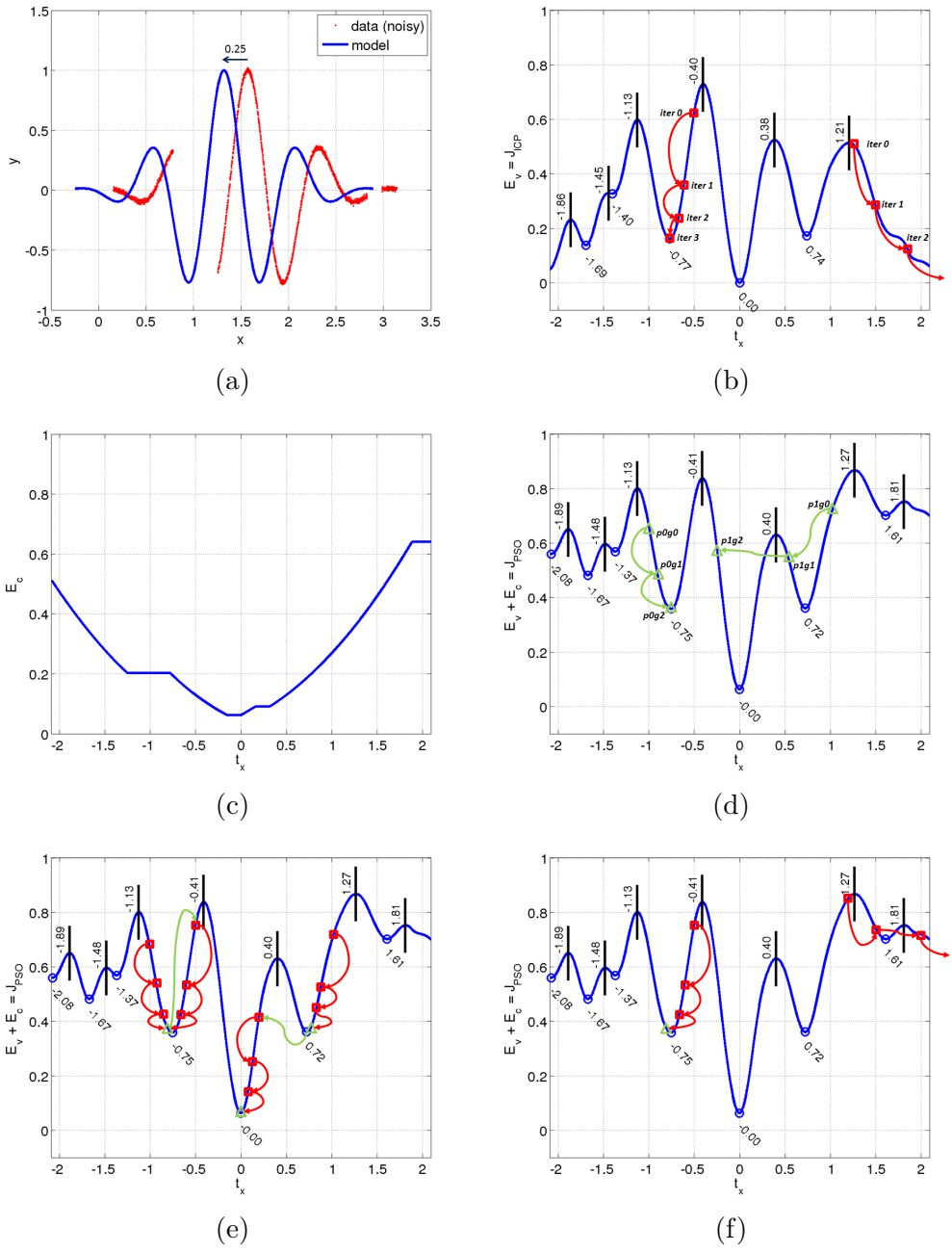


Figure 3.6: (a) A simple 1D registration problem: the reference model (shifted) and noisy sampled data. (b) ICP minimization of $E_v = E_v(t_x)$ (2 runs); t_x is the model shift along the x axis. Local minima are indicated by circles, the boundaries of the basins of attraction by vertical lines. (c) The overlap term $E_c = E_c(t_x)$. Optimization of $E_v(t_x) + E_c(t_x)$ with (d) PSO only, and with (e) the combined PSO and ICP algorithm. (f) ICP moves the left particle to the local minimum of E_v; the right one is moved away from the local basin of attraction of $E_v + E_c$.

CHAPTER 4

FACE VERIFICATION

4.1 Introduction

Face verification and recognition has been an active research topic for the past several decades [48]. Face verification is still a difficult problem due to the wide range of possible head poses and facial expressions. There has been a wealth of 2D face verification methods proposed over the years [49, 50, 51, 52, 53], and there have been some recent advancements using deep convolutional neural networks [54, 55]. However, 2D methods are still not widely used for authentication since they are easily fooled with a photograph. Face verification using 3D sensors has the potential to be a more secure and reliable mode of authentication, and a variety of 3D face verification methods have been proposed [56, 57, 58, 59]. Unfortunately, these methods often require high-quality 3D data captured from a high-end 3D sensor, which is not always practical for real-world situations where noisy low-cost depth cameras are utilized.

We propose a method that uses a consumer-grade depth camera, such as Microsoft's Kinect, to perform online face verification. To be able to verify the identity of a subject, we first construct a reference model by fitting a 3D morphable face model [19]. Given a novel depth image and a reference model, we can authenticate the user by aligning the reference to the image, densely extracting facial features, and computing a similarity metric. To handle a wide variety of situations, we learn our similarity metric using a random decision forest [60]. Since the Kinect can capture depth images at video rate, and our method runs at near real-time rates, we can continuously authenticate a person while he/she uses his/her device.

We evaluate our method using a combination of three datasets. We combine the Biwi Kinect dataset [45] and the Eurecom Kinect dataset [12], as

well as a dataset we collected ourselves. On this hybrid dataset, we demonstrate superior results compared to existing state-of-the-art 2D and 3D face verification methods.

4.2 Related Work

A wide variety of 2D and 3D face verification methods have been proposed over the decades [48].

A large portion of the proposed techniques perform face verification on 2D images [49, 50, 51, 52, 53, 54, 55]. Savvides *et al.* [49] utilized minimum average correlation energy filters for 2D face verification. For each subject, they have a set of 2D images of the subject's face with different expressions, and they used a set of correlation filters to compare the database to a novel image. Jonsson *et al.* [50] represent a 2D face image as a linear combination of basis images, also known as Eigenfaces [61], and trained a support vector machine (SVM) to determine whether or not two face images match. Zhou and Wei [51] represent the face with a set of Gabor wavelet features. They use the AdaBoost algorithm to select the best wavelet features for distinguishing a subject from other subjects, and they train an SVM using these features. Chopra *et al.* [52] utilize a convolution neural network to map a face image to a feature space, such that the Euclidean distance between feature vectors represents how similar two faces are to each other. Kumar *et al.* [53] learned two separate classifiers to perform face verification. One classifier recognizes attributes such as gender, race, age, physique, hair style, eye-wear, etc. The presence or absence of these features is used to verify a person's identity. The second classifier measures the similarity of facial regions. They combine the results of both classifiers to verify the identity of a subject. Taigman *et al.* [54] proposed a similar method to [52], but uses a deep network to map an image to a feature vector, and they train an SVM using these features. Most recently, [55] proposed another approach that uses a deep neural network to learn a Euclidean embedding of face images for recognition, verification, and clustering. Taigman *et al.* [54] and Schroff *et al.* [55] leverage deep convolutional neural networks to sufficiently outperform all previous 2D methods. Face verification techniques that utilize 2D images typically work well under ideal lighting conditions and when the face is viewed from a frontal position,

and their accuracy can degrade when this is not the case. Leveraging 3D data for face verification has the potential to overcome these limitations. 3D sensors often emit their own light, so they are less affected by external lighting conditions. In addition, by leveraging a full 3D model of the face, it is possible for 3D methods to be less influenced by changes in head pose.

Several methods that utilize 3D information have been proposed for face verification [56, 57, 58, 59]. Beumier and Acheroy [56] align the profiles of two 3D facial surfaces, and use the residual error after alignment to compare the similarity of the two faces. In addition, they compare the difference in the intensity images, which were used to reconstruct the geometry of the face. Pan *et al.* [57] use the Hausdorff distance to compare two aligned 3D face models. Conde and Serrano [58] use spin images [62] to detect facial features. The location of the features are used to create a normalized depth image of the face. Principal component analysis (PCA) is used to project the depth images to an eigenvector subspace. An SVM classifier is trained to verify faces within this eigenspace. Lu *et al.* [59] construct a 3D model of a person's face by merging a set of depth images captured by a laser scanner. Given a novel depth image, they align the image to a 3D model using iterative closest point (ICP), and the residual error after alignment is used to verify the subject. Like [56], [59] also considers the 2D appearance of the user by comparing an intensity image to an image synthesized by rendering the 3D model. Previous 3D face verification methods often require high-quality 3D data and/or both 2D and 3D data to accurately identify a user. We propose a method for 3D face verification using only low-quality depth images captured by a consumer-grade camera.

4.3 Method

4.3.1 Model Fitting

Before we can attempt to verify the identity of a subject, we must create a reference model of person for comparison. We create a reference model of the person's face by fitting a 3D morphable face model to a set of depth images. A morphable model consists of an average 3D face shape $\boldsymbol{\mu}$ and a set of 3D face shape bases $\boldsymbol{S} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_M)$ where $M = 199$ [19]. A novel face

shape is produced through a linear combination of the mean face and face bases,

$$\hat{\boldsymbol{S}} = \boldsymbol{\mu} + \sum_{m=1}^{M} \alpha_m \boldsymbol{s}_m = \boldsymbol{\mu} + \boldsymbol{S}\boldsymbol{\alpha}. \tag{4.1}$$

To fit the morphable model we need to identify the coefficients $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_M)$, such that the face shape $\hat{\boldsymbol{S}}$ best matches our subject.

A single depth image is too noisy to accurately fit a model; therefore, we capture a video sequence of the subject's head in a variety of poses. We fit the model to a subset of depth images from the video sequence selected at uniform intervals. For each depth image i, we identify a set of corresponding points $\mathcal{C}_i$ by finding the closest points between the image and the model. We jointly solve for the coefficients of the morphable model $\boldsymbol{\alpha}$ and the pose of each depth image $\{\boldsymbol{R}_i, \boldsymbol{t}_i\}$ by minimizing the following objective function:

$$\min_{\boldsymbol{\alpha}, \{\boldsymbol{R}_i, \boldsymbol{t}_i\}} \sum_i \sum_{j \in \mathcal{C}_i} \left(w_{ij} \left\| (\boldsymbol{R}_i \boldsymbol{v}_{ij} + \boldsymbol{t}_i) - (\boldsymbol{\mu}_{ij} + \boldsymbol{S}_{ij}\boldsymbol{\alpha}) \right\|^2 \right) + \lambda \left\| \boldsymbol{\alpha} \right\|^2, \tag{4.2}$$

where (i, j) is the j-th corresponding point from the i-th image, $\boldsymbol{v}_{ij}$ is a 3D measurement from the i-th depth image, and $\boldsymbol{\mu}_{ij}$ and $\boldsymbol{S}_{ij}$ are the mean and bases of a single vertex from the morphable model corresponding to $\boldsymbol{v}_{ij}$. To reduce the effect of outliers, the point correspondences are weighted inversely proportional to the distance between them, w_{ij}. Also, we use a stiffness term λ to control how much the morphable model is allowed to deform.

We repeatedly solve Eq. (4.2) to refine our estimate of the coefficients $\boldsymbol{\alpha}$ and the poses $\{\boldsymbol{R}_i, \boldsymbol{t}_i\}$. For each iteration, we update our point correspondence between the images and the model. In addition, we will reduce the stiffness term λ if the previous iteration did not significantly change the coefficients. We iterate until λ below a certain threshold.

We use Ceres Solver's [63] implementation of conjugate gradient method to solve Eq. (4.2). For all of our experiments, we use 25 images as a trade-off between quality and computational complexity. Examples reference models are shown in Figure 4.1.

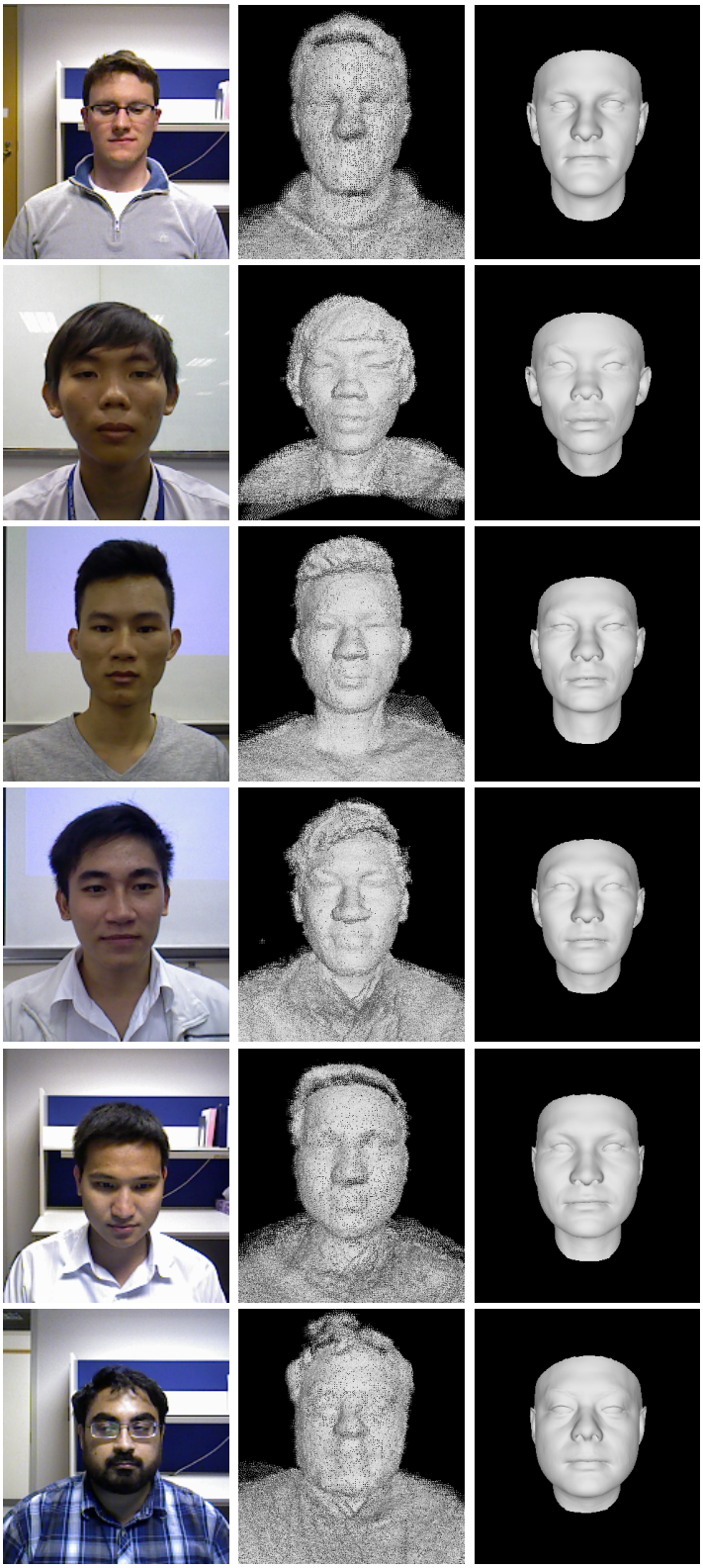


Figure 4.1: Example reference models (right column) constructed by fitting a 3D morphable face model [19] to a point cloud (middle column) assembled from set of depth images. Color images (left column) are shown only for demonstration purposes, and they are not used by our system.

4.3.2 Feature Extraction

To verify whether or not a depth image matches a reference model, we first need to align the model to the image. We use the method described in Chapter 3 to estimate the pose of the head in the depth image. Once we align the reference model to the image, we can extract features which will be used to measure the similarity between the faces. We project every vertex from the reference model into the depth image,

$$\hat{\boldsymbol{v}}_i = \boldsymbol{R}\boldsymbol{v}_i + \boldsymbol{t}, \qquad \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boldsymbol{K}\hat{\boldsymbol{v}}_i, \tag{4.3}$$

where $\boldsymbol{v}_i$ is the i-th vertex in the reference model, $\boldsymbol{R}$ and $\boldsymbol{t}$ are the rotation matrix and translation vector that align the reference model to the image, $\boldsymbol{K}$ is the camera's intrinsic calibration matrix, and (x, y) is the corresponding pixel coordinate in the depth image. Assuming a vertex is not self-occluded and the image is not missing data, we can compute the difference between a vertex and its corresponding depth measurement,

$$\delta_i = \left| D(x, y) - \hat{v}_i^z \right|, \tag{4.4}$$

where $D(x, y)$ is the (x, y)-th measurement in the depth image and $\hat{v}_i^z$ is the z-component of vertex $\hat{\boldsymbol{v}}_i$. Often, measurements are missing from the depth image due to sensor noise and inaccuracies, and model vertices can be self-occluded due to head pose. In these situations, we set the difference value δ_i to positive infinity. To reduce noise in the difference values, we take the average over neighboring vertices,

$$\bar{\delta}_i = \frac{\sum_{k \in \mathcal{N}_i} \delta_k \cdot \mathbf{1}_{\delta_k < \infty}}{\sum_{k \in \mathcal{N}_i} \mathbf{1}_{\delta_k < \infty}}, \tag{4.5}$$

where $\mathcal{N}_i$ is the 1-ring neighborhood of vertex $\boldsymbol{v}_i$ defined by the mesh topology of the reference model and $\mathbf{1}_{\delta_k < \infty}$ is an indicator function. The aggregate of all the difference values forms our feature vector $\boldsymbol{\delta} = (\bar{\delta}_1, \bar{\delta}_2, \ldots, \bar{\delta}_N)$, where the length of the vector is $N = 53490$.

We use a morphable model to create our reference models; therefore, all subjects will have the same set of vertices with the same topology and only

the position of these vertices will change between subjects. If for example vertex $\boldsymbol{v}_j$ is located on the tip of the nose, then δ_j represents the difference between the nose in the model and the nose in the image regardless of the subject in question. This enables us to learn a sophisticated similarity metric to accurately verify the identity of a user, which is our primary motivation for using the morphable model. Figure 4.2 renders the features extracted by our proposed method.

4.3.3 Similarity Metric

There are a variety of ways to compute a similarity metric based on our feature vector $\boldsymbol{\delta}$. The simplest approach is to compute the mean or medium of all the finite elements in $\boldsymbol{\delta}$. However, due to occlusion, changes in pose or expression, and alterations in hair style or eye-wear, this simple metric may not be sufficient to accurately measure the similarity between our reference model and a depth image. Instead, we chose to learn our similarity metric. Several of the elements in our feature vector may contain infinite values due to self-occlusion of the reference model or missing data in the depth image; for this reason, we decided to use a non-linear classifier. Specifically, we chose a random forest classifier [60] for its simplicity to train, and its past success in 3D vision [11, 17].

A random forest classifier consists of an ensemble of decision trees. Each tree in the forest is trained using a set of training examples. The examples are selected randomly from the data set with replacement. A training example consists of a reference model and a depth image, and a label indicating whether or not the model and the image represent the same person. There are significantly more negative examples (the reference model and the depth image are different people) than positive examples (the reference model and the depth image are the same person), and this imbalance can pose a problem for training. Therefore, we re-weight the examples so that positive examples have a p probability of being drawn and negative examples have a chance of $(1 - p)$. A natural choice for p is 0.5, but we experimentally found that $p = 0.25$ produces better results.

To train a decision tree, we begin by extracting all the features from its set of training examples $\mathcal{S} = \{\boldsymbol{\delta}^n\}$. For each non-leaf node in the tree, we

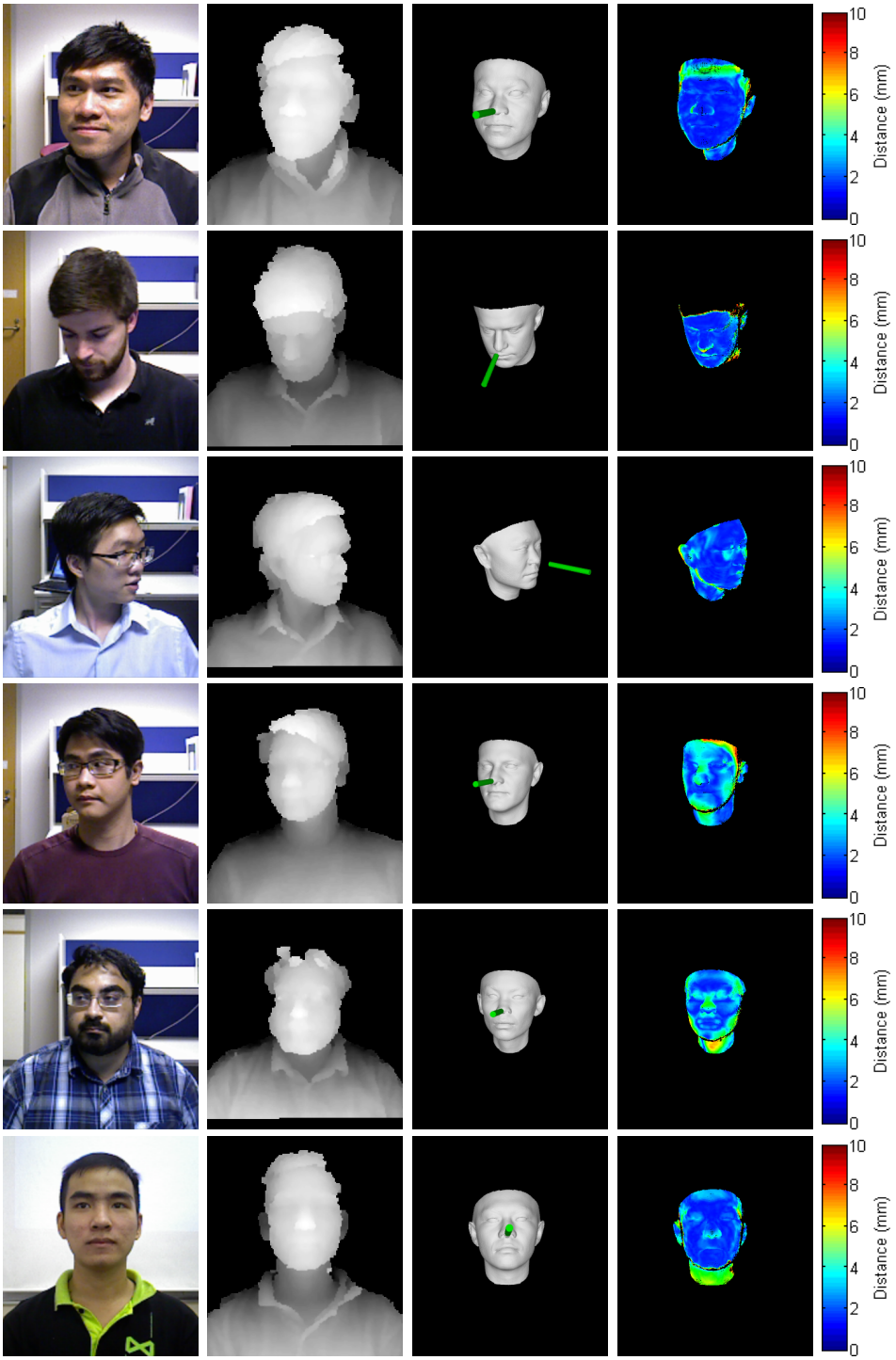


Figure 4.2: Our features (fourth column) are computed by aligning and projecting our reference model (third column) into a depth image (second column) and calculating the difference. These features are used by our learned similarity metric to authenticate a subject. The first three rows depict positive examples where the reference model and the depth image share the same identity. The remaining rows illustrate negative examples. Color images (first column) are only shown for demonstration purposes, and they are not used by our system.

randomly select a subset of features and corresponding thresholds $\{\delta_i, \tau_i\}$, and determine which feature/threshold pair provides the most information gain,

$$\max_{\{\delta_i,\tau_i\}} G(\mathcal{S}', \delta_i, \tau_i) = H(\mathcal{S}') - \left(\frac{|\mathcal{S}_L|}{|\mathcal{S}'|} H(\mathcal{S}_L) + \frac{|\mathcal{S}_R|}{|\mathcal{S}'|} H(\mathcal{S}_R) \right), \tag{4.6}$$

where $H(\cdot)$ is the class uncertainty measure of a set, $|\cdot|$ is the size of a set, $\mathcal{S}' \subset \mathcal{S}$ is the subset of training examples that reach the node, $\mathcal{S}_L = \{\boldsymbol{\delta}^n \mid \delta_i^n < \tau_i,\ \boldsymbol{\delta}^n \in \mathcal{S}'\}$ is the subset of training examples in $\mathcal{S}'$ with feature δ_i less than τ_i, and $\mathcal{S}_R = \mathcal{S}' \setminus \mathcal{S}_L$ [60]. The class uncertainty measurement or entropy of the set is defined as follows:

$$H(\mathcal{S}) = -\left(\frac{|\mathcal{S}|^+}{|\mathcal{S}|} \log_2 \frac{|\mathcal{S}|^+}{|\mathcal{S}|} \right) - \left(\frac{|\mathcal{S}|^-}{|\mathcal{S}|} \log_2 \frac{|\mathcal{S}|^-}{|\mathcal{S}|} \right), \tag{4.7}$$

where $|\cdot|^+$ and $|\cdot|^-$ are the number of positive and negative examples in a set, respectively. When a leaf node is reached, either due to reaching the maximum depth of the tree or too few training examples at the node, then the probability of an positive example reaching this node, $p = |\mathcal{S}'|^+/|\mathcal{S}'|$, is recorded.

To test whether or not a novel example is positive, *i.e.* verify a depth image matches a reference model, we extract features from the example and pass it to each tree in the random forest. At each non-leaf node, we use the stored feature/threshold pair to determine our path through the tree. Once we reach a leaf node in every tree, we average the stored probabilities to produce an overall probability of the image and the model being the same person. If the probability is above a threshold, we claim the image and the model share the same identity.

4.4 Experimental Results

4.4.1 Data Sets

Our goal is to design an online method for face verification using low-quality depth images from a consumer-grade depth camera. Therefore, we require data sets that contain multiple video sequences of a variety of subjects cap-

tured by a Kinect camera or similar device. We measure the performance of our method, as well as that of existing state-of-the-art methods, using a combination of three datasets: the Biwi Kinect dataset [45], the Eurecom Kinect dataset [12], and a dataset we collected ourselves using the Kinect camera. The hybrid dataset contains 102 subjects: 52 from the Eurecom dataset, 20 from the Biwi dataset, and 30 from our own dataset. The dataset comprises 80 males and 22 females with ethnicities including European, Asian, Indian, African, and Hispanic. The majority of the dataset contains at least two video sequences per subject captured during different sessions. For each sequence, the subjects are positioned approximately a meter from the camera. The sequences exhibit a wide range of head poses with only moderate changes in expression. The entire dataset contains over 87K RGB and depth images.

4.4.2 Testing Procedure

For our method and the existing methods we compare against, we use the same procedure to evaluate an approach on the dataset. For each video sequence of a subject, we define a model. The model could be an image, a set of images, a 3D mesh, or some other representation of the subject's identity as specified by the method we are testing. Afterwards, we compare the model to all other images in the dataset, and compute a value based on the method's similarity metric. It is important to note that we do not compare the model to the sequence used to construct it. The output of the method's similarity metric is compared to a threshold to determine its true positive and false positive rate. We vary the threshold to obtain the receiver operating characteristic (ROC) of the approach.

4.4.3 Evaluation of Our Approach

To evaluate our proposed method on the combined dataset, we use 3-fold cross validation. The dataset is randomly partitioned into three equal subsets where each subset contains 34 subjects, and we perform three rounds of training and testing. In each round, two subsets are used for training the random decision forest and the remaining subset is used for testing. Each subset contains approximately 15,000 positive examples and 2,500,000 nega-

tive examples.

For each fold, we trained a random forest of T decision trees each with 50,000 randomly selected training examples to a depth of D. At each non-leaf node we randomly select $(100 \times d)$ feature/threshold pairs where d is the depth of the node. We measure the performance of our method by averaging the results across each fold.

We analyze the behavior of our approach using a variety of different configurations. Figure 4.3 depicts the performance of our method when we vary the number of trees T in the forest and fix the maximum depth to $D = 20$. Beyond 100 trees, we experience diminishing gains in performance. Figure 4.4 shows the results when we fix the number of trees to $T = 100$ and vary the maximum depth D. Allowing the depth of the trees to go beyond 20 levels does not change the output of the random forest because most of the trees do not reach a depth greater than 20 levels. Notice the classifier does not overfit the training data as the number of trees and maximum depth increase; this is a benefit of using the random decision forest [60]. For all the following experiments, we fix the number of trees to $T = 100$ and the maximum depth to $D = 20$.

To support our use of the random decision forest, we compare our approach to a straightforward method where we use the mean or medium of our feature vector $\boldsymbol{\delta}$ as the measure of similarity between the model and the image. Figure 4.5 demonstrates that a significant amount information is lost by performing an average over the feature vector. Further investigation is required to show that the random forest outperforms other types of classifiers.

To gain further insight into the similarity metric learned by the random forest, we count the number of times an element of our feature vector $\boldsymbol{\delta}$ is used by a decision tree to classify an image. The frequencies are illustrated in Figure 4.6. The random forest is learning to focus on the most visually distinctive parts of the face, specifically the nose, brow, chin, and cheeks. Notice, the classifier is capable of learning to prioritize parts of the face that are visible, as we can see in Figure 4.6ac. We believe this is a benefit of using a forest of decision trees because of the many branching paths contained within a tree. In the future, we plan to add dramatic facial expressions to the dataset, and based on these results, we hope the classifier can learn to focus on the non-expressive parts of the face.

For all of our experiments, we perform face verification using a single depth

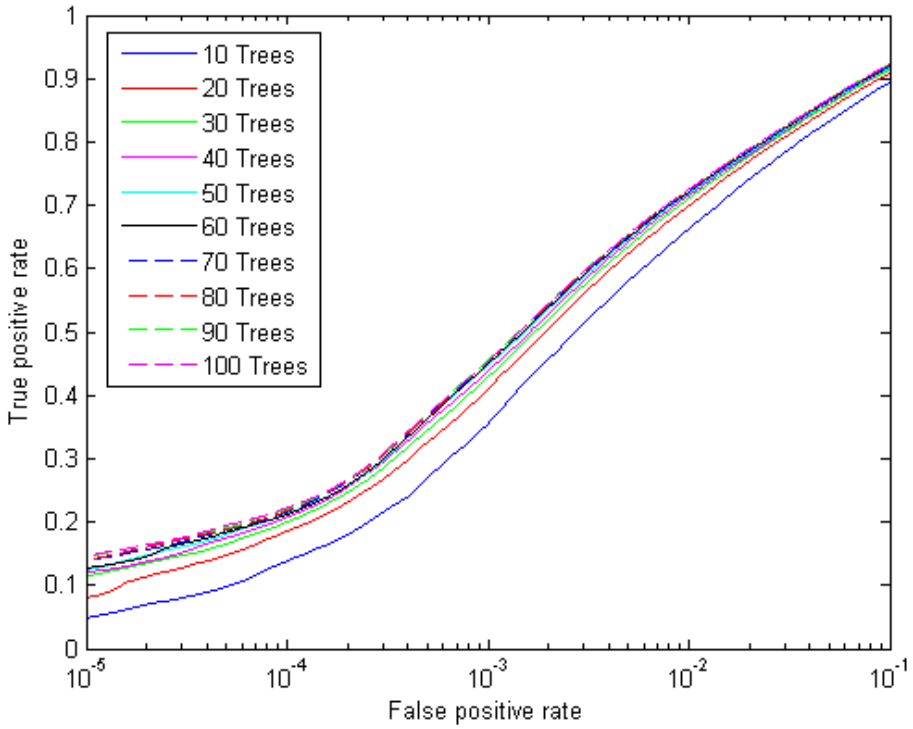


Figure 4.3: The ROC curves of our proposed method when we vary the number of decision trees within the random forest.

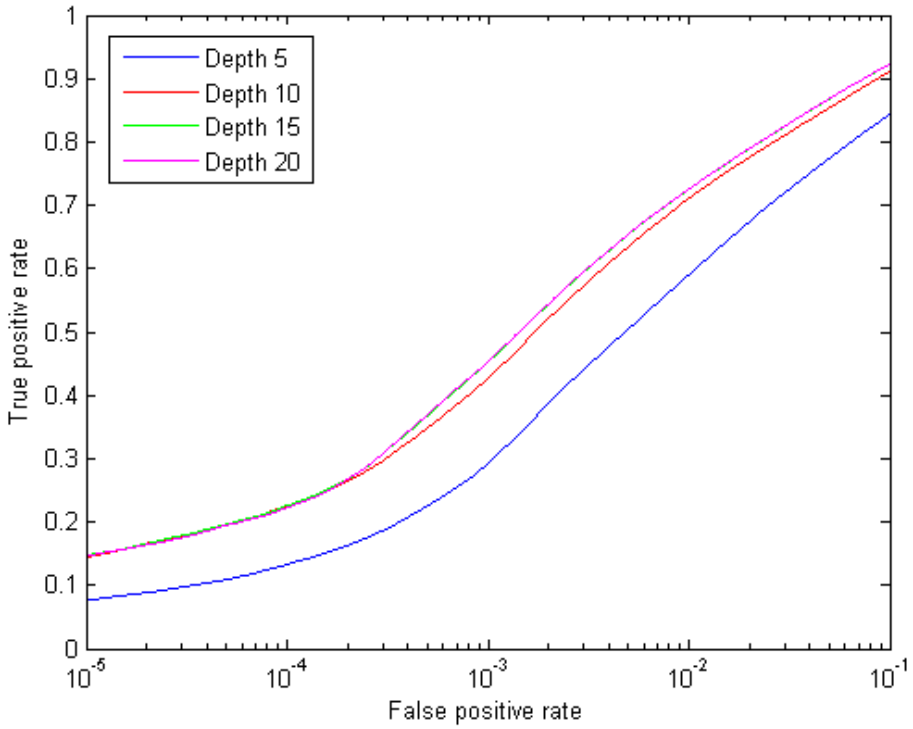


Figure 4.4: The performance of our approach when we change the maximum depth of the decision trees.

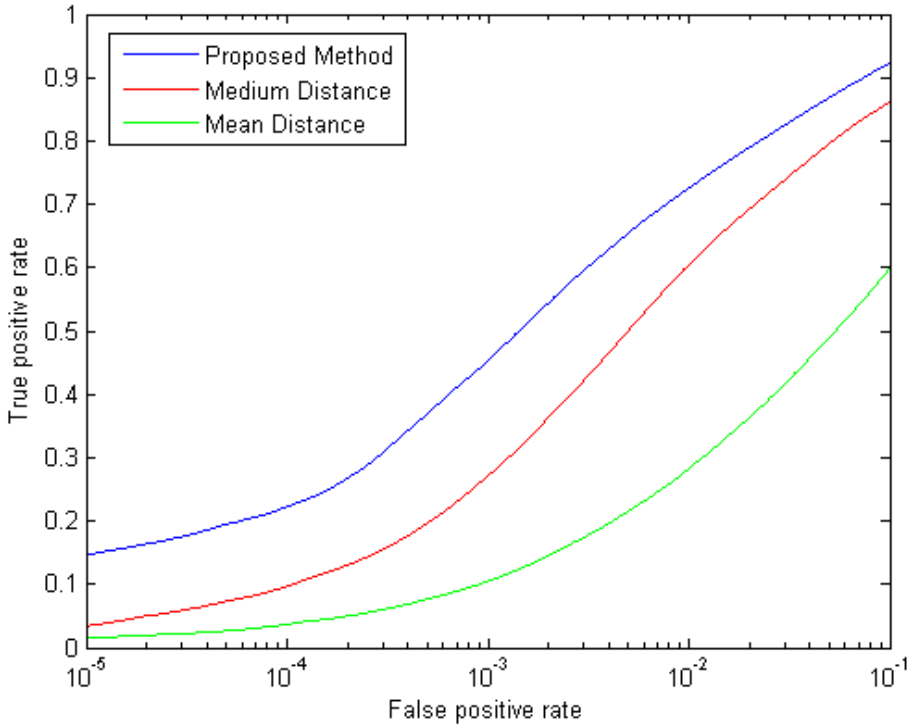
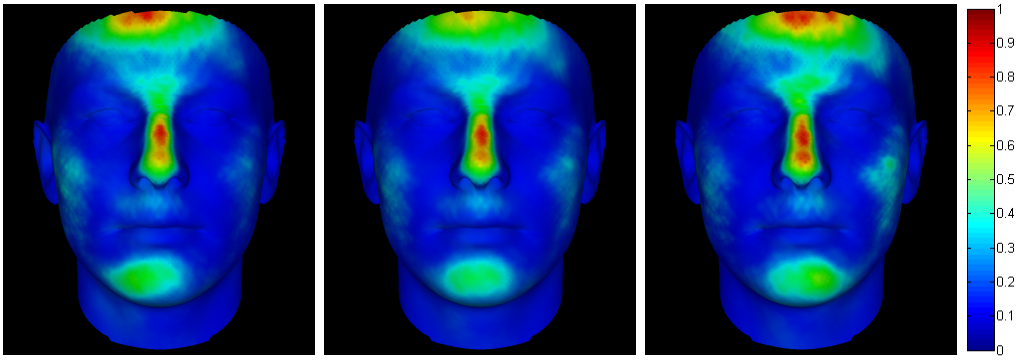


Figure 4.5: Our proposed method compared to two variants where instead of training a random decision forest, we use the mean and medium value of our feature vector $\boldsymbol{\delta}$ to measure the similarity of faces. Clearly, there is a benefit of using the random forest over averaging the feature vector.



(a) (b) (c)

Figure 4.6: A set of heat maps representing the prevalence of the features within the random decision forest. The heat maps show the number of times a feature is used by the random forest to classify an image when the subject is looking (a) left (yaw angle is less than $-30°$), (b) forward (yaw angle is between $-30°$ and $30°$), and (c) right (yaw angle is greater than $30°$). Notice the subtle difference between (a) and (c) where the classifier learned to focus on the side of the face that is visible.

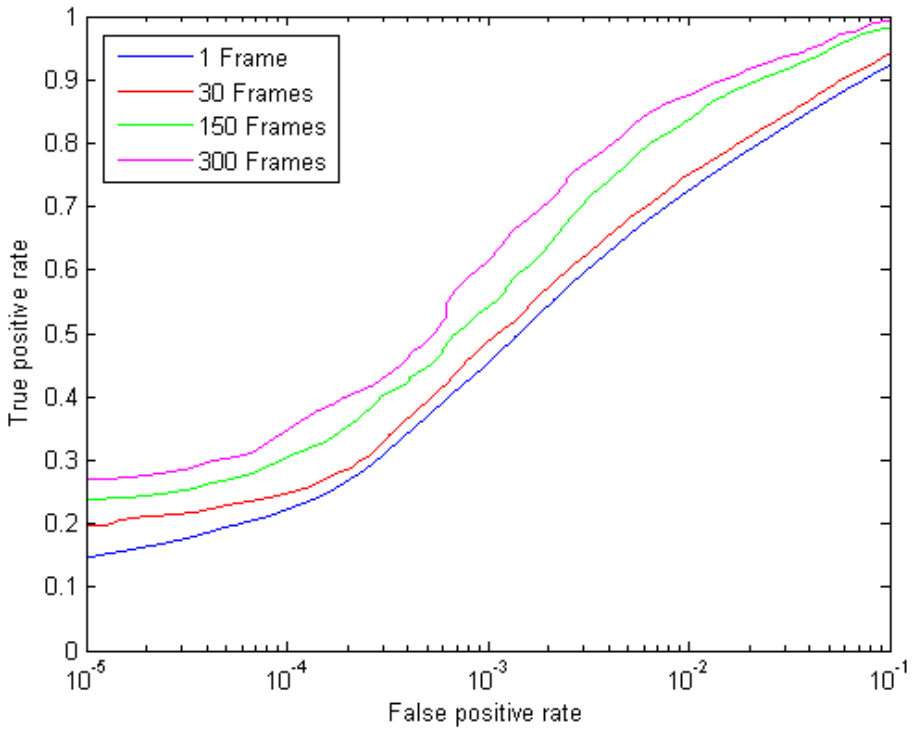


Figure 4.7: The performance of our method when we use a sliding window to combine the decision for a sequence of images. We show the results for when the sliding window contains 30, 150, and 300 frames, which is approximately 1, 5, and 10 seconds of video, respectively.

image; however, it is possible to improve our method's performance by combining the results from a series of images. To this end, we use a sliding window to collect the output of our algorithm for a sequence of images, and the majority decision is used as the decision for entire window. Figure 4.7 shows the results when the sliding window contains 1, 5, and 10 seconds of video where the video is captured at 30 frames per second. These results motivate a possible future extension of our work, where multiple depth images are combined together to improve accuracy. One approach could be to integrate a sequence of depth images into a 3D model like in [64] and [65], and perform verification using this model instead of an image. Although, this approach would require more cooperation by the user.

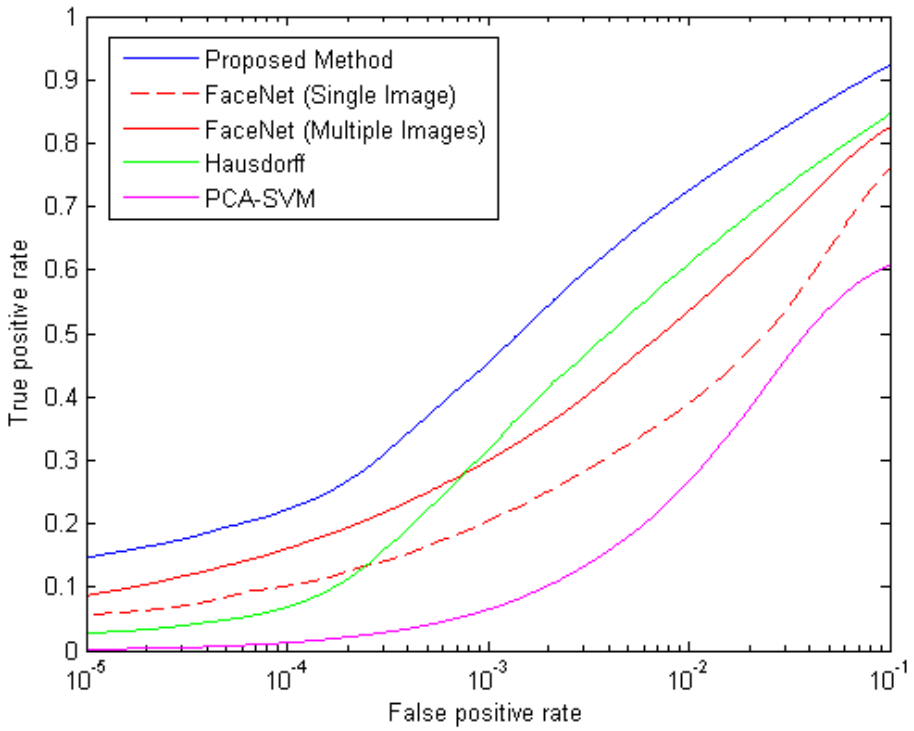


Figure 4.8: Our proposed method compared to existing state-of-the-art 2D and 3D face verification methods [55, 57, 58].

4.4.4 Comparison with Existing Methods

We compare our method to existing state-of-the-art 2D and 3D face verification methods on the hybrid dataset, and the results are shown in Figure 4.8.

For 2D methods, we compare against FaceNet [55]. FaceNet leverages a deep convolutional neural network to map 2D face images to a Euclidean space where distances can be used to measure face similarity [55]. We use OpenFace [66], the open source implementation of FaceNet, to embed each image in the dataset into this feature space. For each sequence, we hand select an image to be used as the subject's reference model. The squared $L2$ distance in the Euclidean space is used as the similarity metric between the model and the dataset images. The performance of FaceNet using a single image is depicted as the dashed red line in Figure 4.8.

Our proposed method uses multiple images from a video sequence to construct the 3D reference model of a subject. In order to fairly compare our method with FaceNet, we replace the single hand selected image with the

same 25 images used to construct our 3D face model. We use the minimum distance between all the reference images and a test image as the measure of similarity. We also experimented with using the mean and medium distance, but the minimum distance performed the best. The performance of FaceNet using multiple images is shown as the solid red line in Figure 4.8.

To examine how our proposed method and FaceNet deal with variations in head pose, we fix the false positive rate to 10^{-2} and compute the true positive rate as a function of yaw and pitch. As shown in Figure 4.9, our proposed method does well across a wide range of head poses, which is a benefit of using a 3D method over a 2D method for face verification.

For 3D methods, we compare against [57] and [58]. Pan *et al.* [57] use the partial Hausdorff distance to measure the similarity of two facial surfaces. To evaluate [57], we compute the partial Hausdorff distance between our 3D face models and the 3D point clouds generated by back-projecting the depth images in the dataset. The results are illustrated as the green line in Figure 4.8.

Conde and Serrano [58] use the estimated head pose to frontalize the depth images. Afterwards, the normalized depth images are converted to Gaussian images, and principle component analysis (PCA) is performed on the entire dataset. Each image is projected onto the vector space spanned by the top K principal components, and Conde *et al.* train a support vector machine (SVM) to distinguish between images of, and not of, the subject within this vector space [58]. In this case, the SVM is both the reference model and similarity metric. For our experiments, we used $K = 2500$ principal components, and the results are shown as the magenta line in Figure 4.8.

Our approach outperforms existing state-of-the-art 2D and 3D face verification methods on the hybrid dataset. We surpass existing 2D approaches by leverages a full 3D face model; as a result, our method is more robust to changes in head pose. We believe the strength of our approach over existing 3D methods is that we measure the difference between every point in the model and the image, and we learn which of these points are important for authentication. Additional work is required to compare our approach against face verification methods that use both 2D and 3D information.

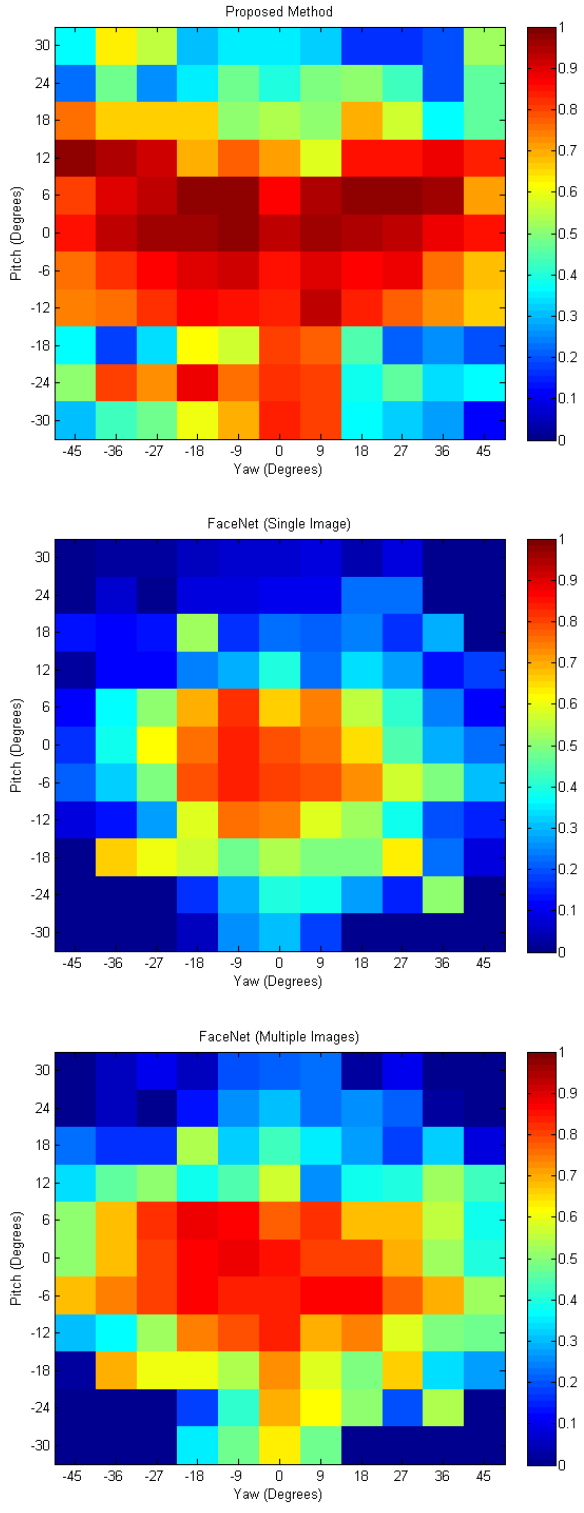


Figure 4.9: The true positive rate as a function of head pose (yaw and pitch) for our proposed method and FaceNet [55]. The false positive rate for each method is fixed to 10^{-2}.

4.4.5 Runtime Performance

Our proposed method is designed to authenticate a user in an online fashion. In other words, when a subject approaches a terminal, our method immediately verifies the user and grants them access. When the subject leaves the terminal or an intruder is detected, our method automatically revokes access. As a result, our system needs to run at near real-time rates.

Model fitting is an offline process, so performance is not a concern. Head pose estimation, feature extraction, and verification need to be run online. Pose estimation is run mostly on the GPU, whereas feature extraction and verification run solely on the CPU. With an Intel Core i7 CPU and NVIDIA GeForce GTX 660 GPU, our system runs at approximately 25 frames per second, which is sufficient for online face verification.

4.5 Concluding Remarks

We proposed an online method for 3D face verification using a low-cost depth camera. We authenticated a subject by comparing a novel depth image to a 3D morphable model which was fit to their face. We trained a random decision forest to learn what facial features are important for measuring the similarity between the image and the reference model. Our proposed method outperforms existing state-of-the-art 2D and 3D methods on a collection of three datasets. Our system runs at 25 frames per second on commodity hardware, so we can continuously verify a stream of depth images as a subject uses his/her device.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, we proposed a system for real-time 3D face localization and verification. Our system consists of three parts: face detection, head pose estimation, and face verification. Each part utilizes depth information from a consumer depth camera.

In Chapter 2, we used the geometrical information within a depth image to identify regions of a color image that may contain a face. As a result, we avoid the exhaustive search over the entire image for faces, which is computationally expensive and prone to false detections. Our approach accelerates the detection process by 3.5x and significantly reduces false detections.

In Chapter 3, we proposed a robust and precise 3D head pose estimation method. Our approach uses a combination of PSO and ICP to register an adaptive 3D face model to a depth image. We demonstrate best-in-class accuracy on standard benchmark datasets.

In Chapter 4, we introduced an accurate method for 3D face verification. Leveraging the techniques proposed in Chapters 2 and 3, we can align and compare a reference 3D face model with a depth image. We learned a similarity metric to determine whether the model and the image share the same identity. Our proposed method exhibits higher accuracy than existing state-of-the-art 2D and 3D methods on a combination of three datasets.

By combining all of these components, we establish a system for real-time 3D face verification using a low-cost depth camera. With our system, we can continuously authenticate a user while he/she is using his/her device.

Several aspects of our work can be improved upon. We plan to explore alternative methods for performing face detection on a depth image, which will enable our system to operate when color information is unavailable.

We would like to evaluate our method when dramatic facial expressions are present. The existing dataset only contains moderate changes in expressions; therefore, we plan to capture additional video sequences with subjects

performing different emotions.

Additionally, we observed that combining the results from a sequence of images improves our system's ability to accurately verify a person's identity. A possible extension of our work could be to explore different methods of combining a sequence of depth images. For example, the depth images could be registered and integrated into a 3D model, and our system could measure the similarity between the reconstructed model and the reference model.

REFERENCES

[1] E. Hjelmas and B. K. Low, "Face detection: A survey," *Computer Vision and Image Understanding*, vol. 83, no. 3, pp. 236 – 274, 2001.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–511.

[3] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[4] M. Dixon, F. Heckel, R. Pless, and W. D. Smart, "Faster and more accurate face detection on mobile robots using geometric constraints," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, Oct 2007, pp. 1041–1046.

[5] H. Wu, K. Suzuki, T. Wada, and Q. Chen, "Accelerating face detection by using depth information," in *Advances in Image and Video Technology*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5414, pp. 657–667.

[6] J. Kovac, P. Peer, and F. Solina, *Human Skin Color Clustering for Face Detection*. IEEE, 2003, vol. 2.

[7] P. De Leva, "Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters," *Journal of Biomechanics*, vol. 29, no. 9, pp. 1223–1230, 1996.

[8] R. M. Haralock and L. G. Shapiro, *Computer and Robot Vision*. Addison-Wesley Longman Publishing Co., Inc., 1991.

[9] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of online learning and an application to boosting," in *European Conference on Computational Learning Theory*. Springer, 1995, pp. 23–37.

[10] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 951–970, 2013.

[11] G. Fanelli, T. Weise, J. Gall, and L. Van Gool, "Real time head pose estimation from consumer depth cameras," in *Pattern Recognition*, 2011, pp. 101–110.

[12] R. Min, N. Kose, and J.-L. Dugelay, "Kinectfacedb: A kinect database for face recognition," *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, vol. 44, no. 11, pp. 1534–1548, Nov 2014.

[13] Itseez, "OpenCV," http://opencv.org/.

[14] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation in computer vision: A survey," *TPAMI*, vol. 31, no. 4, pp. 607–626, 2009.

[15] M. Storer, M. Urschler, and H. Bischof, "3d-mam: 3d morphable appearance model for efficient fine head pose estimation from still images," in *CVPRW*, 2009, pp. 192–199.

[16] Y. Cai, M. Yang, and Z. Li, "Robust head pose estimation using a 3d morphable model," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[17] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Comm. ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[18] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect." in *BMVC*, vol. 1, 2011, p. 3.

[19] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Conf. on Computer Graphics and Interactive Techniques*, 1999, pp. 187–194.

[20] Y. Sun and L. Yin, "Automatic pose estimation of 3d facial models," in *ICPR*, 2008, pp. 1–4.

[21] M. D. Breitenstein, D. Kuettel, T. Weise, L. Van Gool, and H. Pfister, "Real-time face pose estimation from single range images," in *CVPR*, 2008, pp. 1–8.

[22] C. Papazov, T. K. Marks, and M. Jones, "Real-time 3d head pose and facial landmark estimation from depth images using triangular surface patch features," in *CVPR*, 2015, pp. 4722–4730.

[23] E. Seemann, K. Nickel, and R. Stiefelhagen, "Head pose estimation using stereo vision for human-robot interaction," in *AFGR*, 2004, pp. 626–631.

[24] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *CVPR*, 2011, pp. 617–624.

[25] S. Tulyakov, R.-L. Vieriu, S. Semeniuta, and N. Sebe, "Robust real-time extreme head pose estimation," in *ICPR*, 2014, pp. 2263–2268.

[26] T. Baltrusaitis, P. Robinson, and L. Morency, "3d constrained local model for rigid and non-rigid facial tracking," in *CVPR*, 2012, pp. 2610–2617.

[27] A. Rekik, A. Ben-Hamadou, and W. Mahdi, "3d face pose tracking using low quality depth cameras," in *VISAPP*, 2013, pp. 223–228.

[28] Q. Cai, D. Gallup, C. Zhang, and Z. Zhang, "3d deformable face tracking with a commodity depth camera," in *ECCV*. Springer, 2010, pp. 229–242.

[29] C. Cao, Y. Weng, S. Lin, and K. Zhou, "3d shape regression for real-time facial animation," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 41:1–41:10, 2013.

[30] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime performance-based facial animation," *ACM Trans. Graphics*, vol. 30, no. 4, p. 77, 2011.

[31] Y. Zhu and K. Fujimura, "3d head pose estimation with optical flow and depth constraints," in *3-D Digital Imag. and Modeling*, 2003, pp. 211–216.

[32] Y. Wang, X. Huang, C.-S. Lee, S. Zhang, Z. Li, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang, "High resolution acquisition, learning and transfer of dynamic 3-d facial expressions," in *Computer Graphics Forum*, vol. 23, 2004, pp. 677–686.

[33] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz, "Spacetime faces: High-resolution capture for˜ modeling and animation," in *Data-Driven 3D Facial Animation*, 2008, pp. 248–276.

[34] H. Li, J. Yu, Y. Ye, and C. Bregler, "Realtime facial animation with on-the-fly correctives," *ACM Trans. Graph.*, vol. 32, no. 4, p. 42, 2013.

[35] P. Padeleris, X. Zabulis, and A. A. Argyros, "Head pose estimation on depth data based on particle swarm optimization," in *CVPRW*, 2012, pp. 42–49.

[36] T. Bar, J. F. Reuter, and J. Zollner, "Driver head pose and gaze estimation based on multi-template icp 3-d point cloud alignment," in *ITS*, 2012, pp. 1797–1802.

[37] M. Martin, F. Van De Camp, and R. Stiefelhagen, "Real time head model creation and head pose estimation on consumer depth cameras," in *3DV*, vol. 1, 2014, pp. 641–648.

[38] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *Evolutionary Programming VII*, 1998, pp. 601–610.

[39] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *CVPR*, 2014, pp. 1106–1113.

[40] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3d face model for pose and illumination invariant face recognition," in *Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on.* IEEE, 2009, pp. 296–301.

[41] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, 2010, pp. 760–766.

[42] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *Trans. on Evolutionary Comp.*, vol. 6, no. 1, pp. 58–73, 2002.

[43] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Robotics-DL*, 1992, pp. 586–606.

[44] K. L. Low, "Linear least-squares optimization for point-to-plane ICP surface registration," Chapel Hill, University of North Carolina, Tech. Rep. TR04-004, 2004.

[45] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool, "Random forests for real time 3d face analysis," *Int. J. Comp. Vision*, vol. 101, no. 3, pp. 437–458, 2013.

[46] T. Weise, B. Leibe, and L. Van Gool, "Fast 3d scanning with automatic motion compensation," in *CVPR*, 2007, pp. 1–8.

[47] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[48] A. F. Abate, M. Nappi, D. Riccio, and G. Sabatino, "2d and 3d face recognition: A survey," *Pattern Recognition Letters*, vol. 28, no. 14, pp. 1885–1906, 2007.

[49] M. Savvides, B. V. Kumar, and P. Khosla, "Face verification using correlation filters," *3rd IEEE Automatic Identification Advanced Technologies*, pp. 56–61, 2002.

[50] K. Jonsson, J. Matas, J. Kittler, and Y. Li, "Learning support vectors for face verification and recognition," in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on.* IEEE, 2000, pp. 208–213.

[51] M. Zhou and H. Wei, "Face verification using gaborwavelets and adaboost," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 1. IEEE, 2006, pp. 404–407.

[52] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 539–546.

[53] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and simile classifiers for face verification," in *Computer Vision, 2009 IEEE 12th International Conference on.* IEEE, 2009, pp. 365–372.

[54] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on.* IEEE, 2014, pp. 1701–1708.

[55] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[56] C. Beumier and M. Acheroy, "Face verification from 3d and grey level clues," *Pattern Recognition Letters*, vol. 22, no. 12, pp. 1321–1329, 2001.

[57] G. Pan, Z. Wu, and Y. Pan, "Automatic 3d face verification from range data," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 3. IEEE, 2003, pp. III–193.

[58] C. Conde and A. Serrano, "3d facial normalization with spin images and influence of range data calculation over face verification," in *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on.* IEEE, 2005, pp. 115–115.

[59] X. Lu, A. K. Jain, and D. Colbry, "Matching 2.5 d face scans to 3d models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 1, pp. 31–43, 2006.

[60] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[61] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[62] A. E. Johnson, "Spin-images: a representation for 3-d surface matching," Ph.D. dissertation, Carnegie Mellon University, 1997.

[63] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[64] M. Hernandez, J. Choi, and G. Medioni, "Laser scan quality 3-D face modeling using a low-cost depth camera," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, Aug 2012, pp. 1995–1999.

[65] G. P. Meyer and M. N. Do, "Real-time 3d face modeling with a commodity depth camera," in *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2013, pp. 1–4.

[66] B. Amos, B. Ludwiczuk, J. Harkes, P. Pillai, K. Elgazzar, and M. Satyanarayanan, "OpenFace: Face Recognition with Deep Neural Networks," http://github.com/cmusatyalab/openface.